SRS for IdeaStorm

Table of Contents

1 Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.2.1 *Drawing Engine* Scope 4

*1.2.2* *Gallery* Scope 4

1.3 Definitions, Acronyms, and Abbreviations 5

1.4 References 6

1.5 Overview 6

SR003 Sample Requirement 7

2 Overall Description 8

2.1 Product Perspective 8

2.1.1 System Interfaces 8

2.1.2 User Interfaces 8

2.1.3 Operations 8

2.2 Product Functions 9

2.2.1 *Drawing Engine* Functions 9

2.2.2 *Gallery* Functions 9

2.3 User Characteristics 10

2.4 Constraints 10

2.5 Assumptions and Dependencies 10

2.6 Apportioning of Requirements 10

3 Specific Requirements 11

3.1 *Drawing Engine* Requirements 11

3.1.1 External Interface Requirements 11

3.1.1.1 General *Drawing Engine UI* Requirements 11

EIR005 – *Drawing Engine* Rotation (Essential) 11

EIR018 – Display *Drawing Engine* Instructions (Optional) 11

3.1.1.2 *Drawing Engine* Gesture Requirements 11

EIR001 – Dot Drawing Gesture (Essential) 11

EIR002 – Line Drawing Gesture (Essential) 12

3.1.1.3 *Drawing Toolbar* and Buttons Requirements 12

EIR012 – *Drawing* *Toolbar* (Essential) 12

EIR022 – *Quick Switch Button* (Essential) 12

EIR025 – Changing the *Tool Set* (Essential) 12

EIR019 – Hint Button (Optional) 12

EIR011 – New Drawing Button (Essential) 12

EIR021 – Current *Tool Set* Selection (Conditional) 12

EIR028 – *Drawing Tool* Selection (Essential) 13

EIR026 – *Brush* Selection (Essential) 13

EIR027 – *Drawing Color* Selection (Essential) 13

EIR029 – *Gallery* Button (Essential) 13

3.1.2 Functional Requirements 13

3.1.2.1 Manipulating the *Drawing Area* 13

FR001 – Drawing a Dot (Essential) 13

FR002 – Drawing a Line (Essential) 13

FR003 – Erasing the *Drawing Area* (Essential) 13

FR013 – Saving a *Drawing Image* 14

3.1.2.2 Changing the *Tool Set* 14

FR033 – Initial *Tool Set* Selection (Conditional) 14

FR032 – Changeable *Drawing Tool* (Essential) 14

FR004 – Changeable *Brush* (Essential) 14

FR005 – Changeable *Drawing Color* (Essential) 15

FR017 – *Tool Set* Switching (Essential) 15

3.1.2.3 Description of *Drawing Tools* 15

FR020 – Pen *Drawing Tool* (Essential) 15

FR023 – Eraser *Drawing Tool* (Essential) 16

FR021 – Pencil *Drawing Tool* (Conditional) 16

3.1.3 Performance Requirements 16

PR001 – Drawing Response (Essential) 16

PR002 – Erasing Response (Conditional) 16

PR005 – Saving Response (Conditional) 16

3.2 *Gallery* Requirements 17

3.2.1 External Interface Requirements 17

3.2.1.1 General *Gallery* Requirements 17

EIR044 – Display *Gallery Items* (Essential) 17

EIR038 – *Gallery* Rotation (Essential) 17

EIR039 – Scrolling *Gallery* (Essential) 17

3.2.1.2 *Gallery Toolbar* 17

EIR030 – *Gallery Toolbar* Edit Mode (Essential) 17

EIR031 – *Gallery Toolbar* Normal Mode (Essential) 17

EIR032 – Add *Stack* Button (Essential) 18

EIR033 – Add Drawing Button (Essential) 18

EIR034 – Delete Button (Essential) 18

EIR035 – Export Button (Optional) 18

EIR036 – Make *Stack* Button (Conditional) 18

EIR037 – *Gallery* Tutorial Button (Optional) 19

3.2.1.3 Interacting with *Gallery Items* 19

EIR041 – *Gallery Item* Selection (Essential) 19

EIR045 – Open *Gallery Item* (Essential) 19

EIR047 – Moving Up a *Stack Level* (Essential) 19

3.2.2 Functional Requirements 19

FR041 – Initial *Stack* (Essential) 19

FR035 – Add *Stack* (Essential) 19

FR042 – Add *Drawing Object* (Essential) 19

FR036 – Delete *Gallery Item* (Essential) 20

FR037 – Open a *Stack* (Essential) 20

FR014 – Opening a *Drawing Image* 20

FR038 – Convert *Drawing Object* to *Stack* (Conditional) 20

FR040 – Export *Drawing Object* (Optional) 20

3.2.3 Performance Requirements 20

PR006 – Loading Response (Conditional) 20

3.3 Design Constraints 21

3.4 Logical Database Requirements 21

3.5 Other Requirements 22

Annex A – Drawing Data Flow Diagram 22

Annex B – *Command* Data Flow Diagrams 22

Annex C – *Command* Dictionary 23

# Introduction

## Purpose

This document outlines the requirements for the IdeaStorm. This document has been updated to include the Gallery and minor updates to the drawing engine. The audience for this document will be Robert Cole, the Software Engineer making IdeaStorm and Dr. Lohr, his professor. Dr. Lohr is also serving as the substitute mentor for this project due to the current staffing shortage in the UWF Computer Science department.

## Scope

*IdeaStorm* is an *app* for the iPad that allows the *app user* to create multiple drawings in rapid succession for the purpose of brainstorming visually. The *drawing engine* for *IdeaStorm* was created in iteration 2. This iteration will concentrate on creating the *gallery* for *IdeaStorm.* However several modifications will be made to the drawing engine as well.

### *Drawing Engine* Scope

The *drawing engine* for *IdeaStorm* will be enhanced to include the ability to save a *drawing image* from the *drawing area.* The ability to navigate to the *gallery* from the *drawing engine* will also be added.

This version of the *drawing engine* does not provide a means to load previously saved *drawing images* so they can be edited again. Instead once a *drawing image* is saved it can only be viewed in the *gallery*.

### *Gallery* Scope

As part of this iteration, a *gallery* will be added to *IdeaStorm.* This *gallery* will allow the *app user* to navigate through and view previously created *drawing images* and groups of these *drawing images* called *stacks*. This will be provided through a UIScrollView or another interface that is similar to the one implemented in the *Camera Roll.*

The *gallery* will allow additional *drawing images* and additional *stacks* to be added to existing *stacks.* The *gallery* will also provide a means of modifying and deleting *drawing images* and *stacks*.In addition, *drawing images* will be able to be exported to the *Camera Roll*.

The *gallery* will also provide a help option that will display instructions on how to use the various functions.

This version of the gallery will not support the naming or rating of *drawing images* or *stacks.* There will not be any means to duplicate any of these items.

## Definitions, Acronyms, and Abbreviations

* *Active Tool Set* – The *tool set* that is currently being used by the *app user* for drawing.
* *App* – An application that is made for a mobile device.
* *App User* – Someone that will use the *IdeaStorm* *app.*
* *Camera Roll –* This refers to the native iPad app that allows the *app user* to view saved images. This is also known as the Gallery app.
* *Command –* This is an action performed by the user within the system.
* *Database* – The class(es) or module(s) responsible for saving, modifying and retrieving data in the *app*.
* *Drawing Area* – The area on the screen that will respond to touch for drawing purposes.
* *Drawing* *Engine* – The class(es) or module(s) that allow the *app user*  to draw images on the screen using touch.
* *Drawing Step* – A drawing action as defined in FR001 or FR002 that begins when *app user* first touches within the *drawing area* and ends when the *app user* stops touching within the *drawing area*.
* *Drawing Toolbar* – This is the *UI* element that holds and groups all buttons used when for drawing.
* *Gallery* – The class(es) or module(s) that allow the *app user* to view and edit previous images that were made using the *Drawing Engine.*
* *Gallery Toolbar – TBD – Need to write a definition for this.*
* *IdeaStorm* – An *app* that allows the user to visually brainstorm by drawing multiple images in rapid succession.
* *iOS* – Apple’s operating system for mobile devices.
* *OpenGL* – OpenGL ES version 1.1 or 2.0 as implemented on the *iOS* platform.
* *Point Sprite* – This is a point rendered by *OpenGL* that has a *width*, *drawing color* and a *texture* applied to it. *Point sprites* are rendered on the screen as squares the size of their defined width, which is referred to as *point size* throughout the rest of this document including the data dictionary. The *texture* provided to *OpenGL* is then applied over the area of the rendered square.
* *Reserve Tool Set* – A *tool set* that is stored in the *drawing engine* for drawing when the *app user* switches the *tool set.*
* *Stack Level* –A *stack level* describes the relative position of a *gallery item* within a *stack’s* hierarchy.
  + Example: If there is a *stack* (called *stack* A) that has a *gallery* *item* and a *stack* (called *stack* B) within it, then moving the *gallery item* into *stack* B would be considered moving the *gallery item* down a *stack level*. Moving a *gallery item* from *stack* B into *stack* A is considered moving the *gallery item* up a *stack level.*
* *Quick Switch Button* – This is a button that allows the user to quickly change between two different *tool sets*.
* *UI* – Acronym for User Interface.

## References

Cole, R. (2011). *SRS for SHelp*, Retrieved August 31, 2011, not published (copy provided upon request)

Gesture Recognizers. (2011). *Event Handling Guide for iOS: Multitouch Events*, Apple Inc., Retrieved October 1, 2011, from [http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizers/GestureRecognizers.html#//apple\_ref/doc/uid/TP40009541-CH6-SW1](http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizers/GestureRecognizers.html)

Multitouch Events. (2011). *Event Handling Guide for iOS: Multitouch Events*, Apple Inc., Retrieved August 31, 2011, from <http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/MultitouchEvents/MultitouchEvents.html>

Rodgers, E. (2011). *SRS for the FPool System*. SOFTWARE SPEC/IMPL - University of West Florida. Retrieved January 17, 2011, from <http://hostsited2l.uwf.edu/d2l/lms/content/viewer/view.d2l?tId=1030921&d2l_body_type=1&ou=561725> (link no longer works, copy can be provided upon request)

STD830. (1998). *IEEE Recommended Practice for Software Requirements Specifications*. IEEE, STD 830-1998.

UITouch Class Reference (2010). *Data Management: Event Handling*, Apple Inc., Retrieved September 8, 2011, from <http://developer.apple.com/library/ios/#DOCUMENTATION/UIKit/Reference/UITouch_Class/Reference/Reference.html>

Wiegers, K. E. (2003). *Software requirements: practical techniques for gathering and managing requirements throughout the product development cycle* (2nd ed.). Redmond, Wash.: Microsoft Press.

## Overview

The structure of this SRS is organized according to [STD830]. The first section of this SRS is an introduction that contains the purpose, scope, definitions, acronyms, abbreviations, references and this overview.

The second section of this SRS provides an overall description of the software to be developed. This includes interfaces & operations, functions, a description of user characteristics, constraints and assumptions. This section also includes a brief description of future requirements that will be implemented.

The third section of this SRS contains the requirements for this software. These requirements deviate some from [STD830] as they are organized similar to [STD830 Annex A.2] with the exception that the requirements are separated by view instead of by mode.

Requirements in section 3 are all given a unique numbers for tracking through the iterations of this project. These numbers will remain the same for each requirement in each document. However the title of the requirement and the requirement information may change in each document. Also, new requirements may be added, removed or modified during the development of this project. Due to this requirements may appear out of order and have non-sequential numbering.

“Requirements are also assigned a degree of necessity as describe in [STD830 section 4.3.5.2]. These degrees, organized from most necessary to least, are: Essential, Conditional and Optional.” [Cole 4]

Annex A – Drawing Data Flow Diagram provides a data flow diagram that describes the data transformations that occur during the drawing operation.

Annex B – *Command* Data Flow Diagrams provides a data flow diagram that describes the data transformations that occur when sending *commands* throughout the system.

Annex C – *Command* Dictionary provides a dictionary for the commands used in Annex B – *Command* Data Flow Diagrams.

There are several formatting conventions used throughout this document that are outlined below.

An example of requirement formatting is below.

SR003 Sample Requirement

Functional requirements will also use the template below.

**Introduction:**

**Inputs:**

**Processing:**

**Outputs:**

**Errors:**

Example text, pseudo code or example code will be in a block format as in the example below.

Sample example text or pseudo code.

Definitions, acronyms and abbreviations will use formatting as in the example below. Data items defined in section 3.4 will also use this format.

*Sample definition.*

To be done items or items that still require additional work will use the format below.

TBD – This is a sample To Be Done, please ignore me

# Overall Description

## Product Perspective

### System Interfaces

The app delegate will initialize the view controller for the *gallery* and the *drawing engine*. The app delegate will also be responsible for initializing the *database* and creating any connections that are necessary between the database, view controllers and any of their sub-components.

The view controllers will perform all other initialization, setup and work needed for the *drawing engine* and the *gallery*. The app delegate will be responsible for any further initialization, setup and work needed for the *database,* as this resource will be shared between the view controllers.

### User Interfaces

*IdeaStorm* will contain two different views that the *app user* can interact with. These views are the *drawing engine* and the *gallery.*

The *drawing engine* will have a touch interface. This interface will respond to an *app user’s* touches on the screen and will create a *drawing image* in response. This interface is described more in detail in section 3.1.1.

The *gallery* will have a touch interface. This interface will respond to the *app user’s* touches and will allow the *app user* to view, add, delete and export *gallery items.* This interface is described more in details in section 3.2.1.

### Operations

The context diagram in below specifies some of the operations that can be performed by the *app user* with *IdeaStorm* in regards to the *commands* that can be performed. This is further expanded on in Annex B – *Command* Data Flow Diagrams.

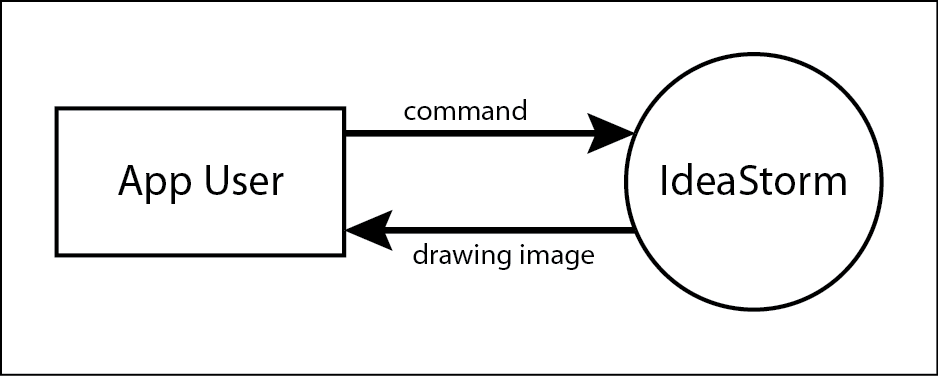


Figure 1 – Context Diagram of IdeaStorm Commands.

The context diagram below specifies the operations that can be performed by the *app user* with the drawing engine. This is further expanded on in Annex A – Drawing Data Flow Diagram.

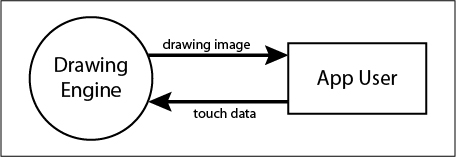


Figure 2 – Context Diagram of Drawing Engine

## Product Functions

### *Drawing Engine* Functions

The *Drawing Engine* will provide the following functions for the *app user.*

1. **Draw on screen:** When the *app user* touches within the *drawing area*, the *drawing engine* will respond by drawing using a *tool set* to draw wherever the *app user* touches. The content rendered in the *drawing area* shall remain on the touch screen when the *app user* stops touching the screen.
2. **Changing the drawing *tool set*:** When the *app user* selects a *drawing tool, drawing color* or *brush* button from the *drawing toolbar*, the *tool set* will be changed to correspond to the *app user’s* selection. This *tool set* change will affect the way that drawing is done when the *app user* touches within the *drawing area*.
3. **Rotating the *Drawing Engine*:** When the *app user* rotates the iPad to a new orientation, all *UI* elements will rotate and change positions to fit the new orientation. The only exception to this is the *drawing area,* which will not change orientation.
4. **New drawing:** When *app user* indicates that they are done with the current drawing, the *drawing engine* will erase the contents of the *drawing area* and provide a blank *drawing area* for the *app user* to start a new drawing in.
5. **Help instructions:** When the *app user* indicates that they need instructions, they will be shown on the screen. The *app user* will be able dismiss and recall these instructions at any time when they are using the *drawing engine.*
6. **Save *drawing image*:** When the *app user* performs the actions for function 4 or 7 on this list, the *drawing engine* will save the current *drawing image*.
7. **Navigate to *Gallery*:** When the *app user* indicates that they would like to view the *gallery*, the *drawing engine* will be dismissed and the *gallery* will be shown.

### *Gallery* Functions

The *Gallery* will provide the following functions for the app user.

1. **View *gallery items:*** The *Gallery* will display thumbnails of all *gallery items* that the *app user* has saved in a hierarchical fashion. The *app user* can view one *stack* at a time. *Stacks* can contain a mixture of *stacks* and *drawing objects*.
2. **Delete *gallery item*:**When the *app user* indicates that they would like to delete a *gallery item,* it will be deleted. If this *gallery item* is a *stack*, all the items that the *stack* contains will also be deleted.
3. **Add *gallery item*:** When the *app user* indicates that they would like to add a new *gallery item*, it will be added to the *stack* they are currently viewing.
4. **Export *drawing object*:** When the *app user* indicates that they would like to export a *drawing object*, the *drawing image* for that *drawing object* will be added to the photo library on their iPad.
5. **Make *stack*:** When the *app user* indicates that they would like to transform a *drawing object* into a *stack*, a *stack* will be created and a *drawing object* will be placed within the new *stack*.
6. **Open *gallery item*:** When the *app user* indicates that they would like to open a *drawing object*, the *drawing image* for the *drawing object* will be displayed in the *gallery.* When the *app user* indicates that they would like to open a *stack*, the contents of that *stack* will be displayed.
7. **Rotating the *Gallery*:** When the *app user* rotates the iPad to a new orientation, all *UI* elements will rotate and change positions to fit the new orientation.

## User Characteristics

The *app user* is assumed have basic knowledge of how an iPad works including: launching an app, interacting with the touch screen, navigating touch interfaces and using gesture interfaces. It is also assumed that the *app user* will download *IdeaStorm* from Apple’s app store.

## Constraints

*IdeaStorm* shall be able to run on the following devices:

* iPad – Running *iOS* version 4.3 or later
* iPad 2 – Running *iOS* version 4.3 or later
* iPad 3 – Running iOS version 5.0 or later

## Assumptions and Dependencies

The developer will integrate the existing *drawing engine* from the previous iteration into *IdeaStorm.* It is also assumed that the developer will create a separate view controller for the *gallery* to allow it to be integrated with the existing *drawing engine.* This is done with the assumption that *IdeaStorm* will have separate view controllers for different functional areas of the *app.*

## Apportioning of Requirements

In this section possible future requirements are described that the *app* developer should keep in mind. These are not to be used as requirements for this version of the *IdeaStorm*; however the developer should keep these in mind when developing the *app.*

* Addition of new *drawing tools*, *brushes*, and *drawing colors* other than those specified in this document.
* The addition of a *drawing color* picker that will allow the *app user* to select any color from the visible spectrum.
* The ability to support in-*app* purchases that include *brush* and/or *drawing tool* packs.
* The ability to undo and redo *drawing steps*.
* The ability to hide and redisplay the *drawing toolbar.*
* The ability to display a preview rendering for *drawing tools.* Example, the straight-line tool will show a preview of the line that it will make once the user stops touching the screen.
* The ability to duplicate *gallery items.*
* The ability to label *gallery items*.
* The ability to load saved *drawing objects* into the *drawing engine* so they can be further edited.
* The ability to reorganize *gallery objects* and move *gallery objects* between *stacks.*

# Specific Requirements

## *Drawing Engine* Requirements

### External Interface Requirements

#### General *Drawing Engine UI* Requirements

This section defines general *UI* requirements that do not fit into any other External Interface Requirements section.

EIR005 – *Drawing Engine* Rotation (Essential)

The *drawing engine* will rotate and reposition all *UI* elements to fit each orientation of the iPad. This includes that all buttons, images and text should be rotated to an upright orientation for the *app user* as long as the rotation lock function is not engaged on the iPad. This includes that all *UI* elements will be repositioned to fit within the viewable screen area. This also includes that all direction dependent gestures should be reoriented to the to the *app user’s* view. The exception for this is the *drawing area* will not rotate and will be kept stationary relative to the iPad.

EIR018 – Display *Drawing Engine* Instructions (Optional)

When the *app user* launches the *drawing engine* for the first time, a set of instructions will be displayed. These instructions will explain the purpose and use of each button, *UI* element and gesture that can be used in the *drawing engine*. When the *app user* is done reading these directions they will be able to dismiss them. The method for dismissing these instructions is left to the developer.

#### *Drawing Engine* Gesture Requirements

This section describes all gestures that will be used in the *drawing engine.*

EIR001 – Dot Drawing Gesture (Essential)

The *app user* will initiate drawing a dot by touching anywhere in the *drawing area* a single time.

EIR002 – Line Drawing Gesture (Essential)

The *app user* will initiate drawing a line by touching anywhere in the *drawing area* and moving their finger across the surface of the *drawing area*.

#### *Drawing Toolbar* and Buttons Requirements

This section describes the *drawing toolbar* and all buttons used in the *drawing engine*.

EIR012 – *Drawing* *Toolbar* (Essential)

The below figure illustrates the basic layout of the *drawing* *toolbar*. The buttons in this *drawing toolbar* are to remain in this general layout with the *quick switch button* in the bottom left corner of the iPad and all other buttons to the right of the *quick switch button* along the bottom of the screen. Any part of the *drawing toolbar* layout not specified here or in other parts of this document is left to the discretion of the developer.

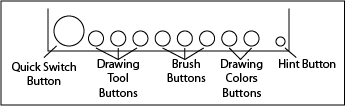


Figure 3 – Drawing toolbar Layout

EIR022 – *Quick Switch Button* (Essential)

When the *quick switch button* is pressed, the *drawing engine* will make the *reserve tool set* the *active tool set* and will make the *active tool set* the *reserve tool set*.

EIR025 – Changing the *Tool Set* (Essential)

When the *drawing tool*, *brush* or *drawing color* buttons are pressed, the *drawing engine* will change the corresponding item in the *active tool set* for the *drawing engine*.

EIR019 – Hint Button (Optional)

When the hint button is pressed, the *drawing engine* instructions will be displayed as described in EIR018.

EIR011 – New Drawing Button (Essential)

The *app user* will be able to start a new drawing by pressing the new drawing button once. This action will first save the current drawing image as described in FR013, and then the *drawing area* will be erased. The *active tool set* and *reserve tool set* remain the same during this process. The placement of this button is left up to the *app* developer with the condition that it remains at the edge of the *drawing area* or is placed in the *drawing toolbar*.

EIR021 – Current *Tool Set* Selection (Conditional)

The *drawing engine* will provide a visual indication of the *brush*, *drawing* *color* and *drawing* *tool* selected in each *active tool set*. This visual indication is left up to the developer.

EIR028 – *Drawing Tool* Selection (Essential)

The *drawing engine* will provide the selection of *drawing tools* in the *drawing toolbar* that are described in section 3.1.2.3.

EIR026 – *Brush* Selection (Essential)

The *drawing engine* will provide at minimum 3 different *brushes* in the *drawing toolbar* that each have a different *texture*. The choice of the *textures* used by the *brushes* is left to the developer.

EIR027 – *Drawing Color* Selection (Essential)

The *drawing engine* will provide at minimum 4 different *drawing colors* in the *drawing toolbar*. These *drawing colors* must include black, red, green and blue defined as [0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 1.0] and [0.0, 0.0, 1.0, 1.0] respectively. Any additional *drawing colors* provided are left to the developer’s discretion.

EIR029 – *Gallery* Button (Essential)

The *app user* will be able to navigate to the *gallery* portion of the app by pressing the *gallery* button once. This will save the current *drawing image* as specified in FR013. The placement of this button is left up to the *app* developer with the condition that it remains at the edge of the *drawing area* or is placed in the *drawing toolbar*.

### Functional Requirements

#### Manipulating the *Drawing Area*

This section describes requirements related to manipulating the *drawing area*.

FR001 – Drawing a Dot (Essential)

**Introduction:** When given the appropriate gesture as defined in EIR001, the *drawing engine* will render a dot in the *drawing area* using *active tool set*.

**Inputs:** Gesture as defined in EIR001.

**Outputs:** A dot made of a single *point sprite*.

FR002 – Drawing a Line (Essential)

**Introduction:** When given the appropriate gesture as defined in EIR002, the *drawing engine* will render a line in the *drawing area* using the *active tool set*.

**Outputs:** A line made of a multiple *point sprites.*

FR003 – Erasing the *Drawing Area* (Essential)

**Introduction:** When appropriate action is received as described in EIR011, the *drawing engine* will clear the *drawing area* of all *app user* drawn content.

**Inputs:** Action as defined in EIR011.

**Processing:** The *drawing engine* will erase the any information regarding the undo or redo operations from the previous drawing. The *drawing engine* will clear the *drawing area* to a white *drawing color* defined as [1.0, 1.0, 1.0, 1.0].

**Outputs:** A blank *drawing area.*

FR013 – Saving a *Drawing Image*

**Introduction:** When the appropriate action is received as described in EIR011 or RIR029, the *drawing engine* will save the *drawing image* as a *drawing object.*

**Inputs:** Action as described in EIR011 or EIR029.

**Processing:** The *drawing engine* will save the current *drawing image* displayed in the *drawing area*. This will not include saving the rendered *drawing toolbar* or any of the other buttons.

**Outputs:**A*drawing object*.

**Errors:** Disk full.

#### Changing the *Tool Set*

This section describes requirements that are related to changing the items stored in the *active tool set*. This section also describes the management of the *active tool set* and the *reserve tool set*.

FR033 – Initial *Tool Set* Selection (Conditional)

**Introduction:** When the *app user* starts the *drawing engine* for the first time, the *active tool set* and the *reserve tool set* will both default to the following: the pen *drawing tool* specified in FR020, a default *brush* chosen by the developer, the black *drawing color* defined in EIR027, and a *point size* set at a default value that is left to the discretion of the developer.

**Processing:** The *active tool set* and *reserve tool set* will have its *drawing tool* set to the pen *drawing tool* specified in FR020, its *brush* will be set to the default *brush*, its *drawing color* will be set to black and its *point size* set to the default value.

**Outputs:** The *drawing engine* will provide the output specified in EIR021.

FR032 – Changeable *Drawing Tool* (Essential)

**Introduction:** When given the appropriate action as defined in EIR025, the *drawing engine* will change the *drawing tool* in the *active tool set* to correspond with the *drawing tool* button pressed by the *app user*.

**Inputs:** Action as defined in EIR025, specifically when a *drawing tool* button is pressed.

**Processing:** The *drawing engine* will update the *drawing tool* in the *active tool set* to correspond with the *drawing tool* button selected.

**Outputs:** The *drawing* engine will provide the output specified in EIR021.

**Errors:** If the *app user* presses the *drawing tool* button that corresponds to the *drawing tool* that is already in the *active tool set*, the *drawing engine* will ignore the action.

FR004 – Changeable *Brush* (Essential)

**Introduction:** When given the appropriate action as defined in EIR025, the *drawing engine* will change the *brush* in the *active tool set* to correspond with the *brush* button pressed by the *app user*.

**Inputs:** Action as defined in EIR025, specifically when a *brush* button is pressed.

**Processing:** The *drawing engine* will update the *brush* in the *active tool set* to correspond with the *brush* button selected.

**Outputs:** The *drawing* engine will provide the output specified in EIR021.

**Errors:** If the *app user* presses the *brush* button that corresponds to the *brush* that is already in the *active tool set*, the *drawing engine* will ignore the action.

FR005 – Changeable *Drawing Color* (Essential)

**Introduction:** When given the appropriate action as defined in EIR025, the *drawing engine* will change the *drawing color* in the *active tool set* to correspond with the *drawing color* button pressed by the *app user*.

**Inputs:** Action as defined in EIR025, specifically when a *brush* button is pressed.

**Processing:** The *drawing engine* will update the *drawing color* in the *active tool set* to correspond with the *drawing color* button selected.

**Outputs:** The *drawing* engine will provide the output specified in EIR021.

**Errors:** If the *app user* presses the *drawing color* button that corresponds to the *drawing color* that is already in the *active tool set*, the *drawing engine* will ignore the action.

FR017 – *Tool Set* Switching (Essential)

**Introduction:** When given the appropriate action as defined in EIR022, the *drawing engine* will switch the *active tool set* and the *reserve tool set*. After this operation the *active tool set* will match what the *reserve tool set* was and the *reserve tool set* will match what the *active tool set* was.

**Inputs:** Action as defined in EIR022.

**Processing:** The *drawing engine* will switch the *tool sets* stored in the *active tool set* and the *reserve tool set*.

**Outputs:** The *drawing* engine will provide the output specified in EIR021.

#### Description of *Drawing Tools*

This section describes the *drawing tools* that are available for use by the *app user* when drawing.

FR020 – Pen *Drawing Tool* (Essential)

**Introduction:** When provided with an array of *positions*, the *drawing color* in the *active tool set* and a *point size*, the pen *drawing tool* will create a *vertices* array. When the *app user* draws on the screen using the pen *drawing tool*, they can create a dot, straight line or curved line that has a consistent width and uses the *drawing color* for its color and the *brush* for its *texture* from the *active tool set*.

**Inputs:** *Position* data*, drawing color* & *point size.*

**Processing:** All *vertex* data created will use the given *drawing color* and *point size* as is. If only one *position* is received, a single *vertex* will be created using this *position* and placed into a *vertices* array. If two *positions* are received, additional *positions* are interpolated along a straight line between these *positions*. All *positions* are then put into individual *vertex* data structures, which are placed in a *vertices* array. If three or more *positions* are received, additional *positions* are interpolated along a Bezier curve path using these *positions*. All *positions* are then put into individual *vertex* data structures, which are placed in a *vertices* array.

**Outputs:** *Vertices*.

FR023 – Eraser *Drawing Tool* (Essential)

**Introduction:** This functions as FR020, with the exception that the *drawing color* is ignored during processing and white is used instead.

**Inputs:** *Position* data*, drawing color* & *point size.*

**Processing:** The processing is handled the same as FR020, except that the *drawing color* provided is ignored and a *drawing color* of white, defined as [1.0, 1.0, 1.0, 1.0], is used to create each *vertex*.

**Outputs:** *Vertices*.

FR021 – Pencil *Drawing Tool* (Conditional)

**Introduction:** This functions as FR020; with the exception that the lines produced with the pencil *drawing tool* will have widths that vary depending on the speed of the drawing.

**Inputs:** *Position* data*, drawing color* & *point size.*

**Processing:** The processing is handled the same as FR020, except that the *point size* provided will be used as a maximum *point size*. As the *positions* provided increase in distance away from each other, each *vertex* will be created with a reduced *point size*. As the distance between *positions* decreases, each *vertex* will be created with an increased *point size* that will not exceed the maximum *point size*. The amount to decrease and increase the *point sizes* in the produced *vertex* data and the threshold for distance to produce a *vertex* with the maximum *point size* is left to the discretion of the developer.

**Outputs:** *Vertices*.

### Performance Requirements

PR001 – Drawing Response (Essential)

When drawing a dot or line as described in FR001 and FR002, the *drawing engine* will display its output within .25 seconds of receiving the gesture described in EIR001 or EIR002. The *drawing engine* will also continue to update the output in response to the *app user’s* touch as described in EIR002 within .25 seconds as the *app user* is dragging their finger across the screen.

PR002 – Erasing Response (Conditional)

When erasing the screen as defined in FR003, the *drawing engine* will clear all *app user* drawn content from the *drawing area* within .25 seconds of receiving the action as described in EIR011.

PR005 – Saving Response (Conditional)

When saving a *drawing image* as defined in FR013, the *drawing engine* will complete the save within .25 seconds.

## *Gallery* Requirements

### External Interface Requirements

#### General *Gallery* Requirements

This section describes the general *UI* requirements for the *gallery.*

EIR044 – Display *Gallery Items* (Essential)

The *gallery* will display previously saved *gallery items* to the *app user*.

EIR038 – *Gallery* Rotation (Essential)

The *gallery* will rotate and reposition all *UI* elements to fit each new orientation of the iPad. This includes that the *gallery toolbar* and the *gallery items* displayed should be rotated to an upright orientation for the *app user* as long as the rotation lock function is not engaged on the iPad. This includes that all *UI* elements will be repositioned to fit within the viewable screen area. This also includes that all direction dependent gestures should be reoriented to the to the *app user’s* view.

EIR039 – Scrolling *Gallery* (Essential)

The *gallery* will allow all content other then the *gallery toolbar* to be scrolled. This scrolling should use the UIScrollView class or should implement another class that will give the gallery functionality similar to the *camera roll*.

#### *Gallery Toolbar*

This section describes the *gallery toolbar* and all buttons used in the *gallery.*

EIR030 – *Gallery Toolbar* Edit Mode (Essential)

The *app user* will be able to switch the *gallery toolbar* into edit mode by tapping the Edit button. In this mode, the *gallery toolbar* will display the Delete (EIR034), Export (EIR035) and Make *Stack* (EIR036) buttons. This is shown in the figure below. During this mode the *app user* will also be able to select *gallery items* (EIR041) and move *gallery items* (EIR042).

EIR031 – *Gallery Toolbar* Normal Mode (Essential)

The *app user* will be able to switch the *gallery toolbar* into a normal mode by tapping the Done button. In this mode, the *gallery toolbar* will display the Add *Stack* (EIR032) and Add Drawing (EIR033) buttons. This is shown in the figure below. During this mode the *app user* will also be able to open *gallery items* (EIR042).

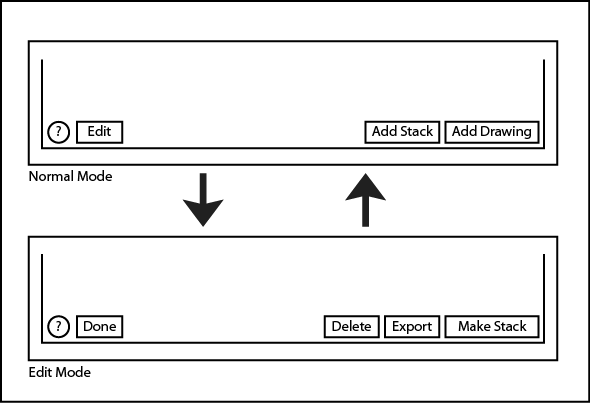


Figure 4 – Gallery Toolbar Normal and Edit Modes

EIR032 – Add *Stack* Button (Essential)

When the *app user* taps the Add *Stack* button, a new stack will be added to the current *stack level*. This will also switch the *app* to the *drawing engine*. The user will then be able to create a series of *drawing images* that will be added to the new *stack* as *drawing objects.*

EIR033 – Add Drawing Button (Essential)

When the *app user* taps the Add Drawingbutton, the *app* will switch to the *drawing engine*. The user will then be able to create a series of *drawing images* that will be added to the current *stack level*.

EIR034 – Delete Button (Essential)

If no selection is currently made the Delete button will be inactive. When the *app user* has selected a *gallery item*, the Delete button will become active. When the *app user* taps the Delete button, the selected *gallery item* will be deleted. This includes that all sub-*gallery items* will be deleted as well. A warning dialogue should appear asking the *app user* to confirm the deletion before it occurs.

EIR035 – Export Button (Optional)

If no selection is currently made the Export button will be inactive. When the *app user* has selected a *gallery item*, the Export button will become active. When the *app user* taps the Export button, the selected *gallery item* will be exported to the camera roll, including all sub-*gallery items.*

EIR036 – Make *Stack* Button (Conditional)

If no *drawing object* selection is currently made the Make *Stack* button will be inactive. When the *app user* has selected a *drawing object*, the Make *Stack* button will become active. When the *app user* taps the Make *Stack* button, a new *stack* will be created at the current *stack level* and the selected *drawing object* will be placed within that stack.

EIR037 – *Gallery* Tutorial Button (Optional)

When the *gallery* tutorial button is pressed, instructions will be displayed that describes how to use the *gallery*.

#### Interacting with *Gallery Items*

This section describes the various interactions that the *app user* may have with the *gallery items* displayed within the *gallery.*

EIR041 – *Gallery Item* Selection (Essential)

When the *app user* taps a *gallery item* that item will be considered selected. The *gallery* will provide a visual indication that the *gallery item* is selected.

EIR045 – Open *Gallery Item* (Essential)

When the *app user* taps a *gallery item* it will open that item. For *drawing objects*, this will open its *drawing image* in the *gallery*. For *stacks*, this will display the contents of the *stack*.

EIR047 – Moving Up a *Stack Level* (Essential)

An icon will be displayed that will represent moving up a *stack level*. When this icon is tapped, the *gallery* will display the contents of the *stack* one *stack level* higher then the current level as long as this *stack level* exists. An example of the placement of this icon is shown in the figure within **Error! Reference source not found.** as item A

### Functional Requirements

FR041 – Initial *Stack* (Essential)

**Introduction:** The *gallery* will create a *stack* when the *app* launches for the first time. This *stack* will be the top *stack level* and will be what all other *gallery items* are stored in.

**Processing:** The gallery will create a *stack* the first time the *app* is launched.

**Outputs:** *Stack*.

**Errors:** Disk full.

FR035 – Add *Stack* (Essential)

**Introduction:** When the appropriate action is received as specified in EIR032, the *gallery* will create a new *stack*, then open that *stack* as specified in FR037 and then add a *drawing object* as specified in FR042.

**Inputs:** Action as specified in EIR032.

**Processing:** Create a new *stack* at the current *stack level*, then open that *stack* as specified in FR037 and then add a *drawing object* as specified in FR042.

FR042 – Add *Drawing Object* (Essential)

**Introduction:** When the appropriate action is received as specified in EIR033, the *gallery* will switch to the *drawing engine*. Any *drawing images* saved during the drawing session will be saved as *drawing objects* within the *stack* that is currently being viewed in the *gallery*.

**Inputs:** Action as specified in EIR033.

**Processing:** Switch to the *drawing engine.*

**Outputs:** *Drawing object(s).*

FR036 – Delete *Gallery Item* (Essential)

**Introduction:** When the appropriate action is received as specified in EIR034, the *gallery* will delete the specified *gallery item*.

**Inputs:** G*allery item.*

**Processing:** Delete *gallery item.*

FR037 – Open a *Stack* (Essential)

**Introduction:** When the appropriate action is received as specified in EIR045, the *gallery* will display the contents of the *stack* specified.

**Inputs:** S*tack.*

**Processing:** Display the contents of the *stack*.

FR014 – Opening a *Drawing Image*

**Introduction:** When the appropriate action is received as described in EIR045, the *gallery* will display the *drawing image* using the provided *drawing object.*

**Inputs:** The *drawing object*.

**Processing:** The *gallery* will display the *drawing image* that is provided as part of the *drawing object*.

**Outputs:** The *drawing image*.

**Errors:** *Drawing image* not found.

FR038 – Convert *Drawing Object* to *Stack* (Conditional)

**Introduction:** When the appropriate action is received as specified in EIR036, the *gallery* will create a new *stack* and place the specified *drawing object* into the *stack*.

**Inputs:** *Drawing object.*

**Processing:** Create *stack* and place *drawing object* into created *stack*.

**Outputs:** *Stack*.

FR040 – Export *Drawing Object* (Optional)

**Introduction:** When the appropriate action is received as specified in EIR035, the *gallery* will get the *drawing image* from the specified *drawing object* and copy it into the *camera roll*.

**Inputs:** *Drawing object.*

**Processing:** Copy the *drawing image* of the specified *drawing object* into the *camera roll.*

**Outputs:** *Drawing image.*

### Performance Requirements

PR006 – Loading Response (Conditional)

When loading a *drawing image* as defined in FR014, the *gallery* will complete the load within .50 seconds.

## Design Constraints

There are no other design constraints other than those listed in sections 2.1.1, 2.4 and 2.5.

## Logical Database Requirements

This section contains a data dictionary for items that will be used in the development of the *IdeaStorm* *drawing engine.* Data dictionaries are described in [Wiegers Chapter 10] and an example of its implementation is found in [Rodgers Section 3.4].

|  |  |
| --- | --- |
| Data Item | Description and Composition (if any) |
| *Brush* | A *texture* used for drawing on the screen.  *texture* |
| *Drawing Color* | A color that is used by a *drawing tool* to draw on the screen.  {*red value [0.0-1.0], blue value [0.0-1.0], green value [0.0-1.0], alpha value [0.0-1.0]*} |
| *Drawing Image* | The image rendered to the screen when the *app user* draws. |
| *Drawing Object* | A type of *gallery item* that contains a *drawing image*.  *drawing image* |
| *Drawing Tool* | An object that receives a *point size, drawing color* and multiple *positions* from the *drawing engine* and transforms them into a set of *vertices*. |
| *Gallery Item* | An object that is displayed in the *gallery.* |
| *Point Size* | The width that is applied to a *point sprite* during rendering. This is a non-zero, non-negative floating point number.  {*width value [>0.0]*} |
| *Position* | A location as described in two-dimensional space using the x-axis and y-axis. This is described by two floating point numbers.  {*xPosition [>=0.0>=]*, *yPosition [>=0.0>=]*} |
| *Stack* | A type of *gallery item* that contains other *gallery items.*  *gallery item + gallery item + gallery item…* |
| *Texture* | An image used by *OpenGL* for rendering a *point sprite* on the screen.  {*image.png*} |
| *Tool Set* | A *drawing tool, brush, drawing color* and *point size* used for drawing by the *app user.*  *drawing tool* + *drawing color* + *brush* + *point size* |
| *Touch Data* | This is the data that the iPad receives when the *app user* touches the screen to interact with the *app.* This is defined in detail in [UITouch]. |
| *Vertex* | A data item provided to the *OpenGL* that includes the *position, drawing color* and *point size* to use when rendering a *point sprite*.  *position* + *drawing color* + *point size* |
| *Vertices* | An array of one or more *vertex* data items.  *vertex* + *vertex* + *vertex…* |

## Other Requirements

There are no other requirements besides those listed in the rest of section 3.

# Annex A – Drawing Data Flow Diagram

The diagram in this annex describes the data transformations that take place during the draw on screen function described in section 2.2. Connections with other parts of the system and error handling are omitted to keep the diagram simple.

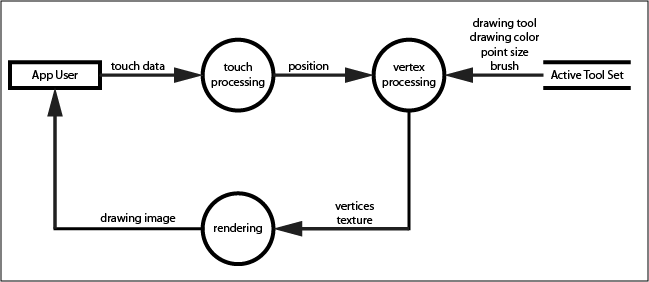


Figure 5 – Drawing Data Flow Diagram

# Annex B – *Command* Data Flow Diagrams

The diagrams in this annex describe the data transformations that take place as the using is issuing *commands* to the *Gallery* and the *Drawing Engine*. Error handling and transformations that are not part of *command* handling are omitted to keep the diagrams simple.

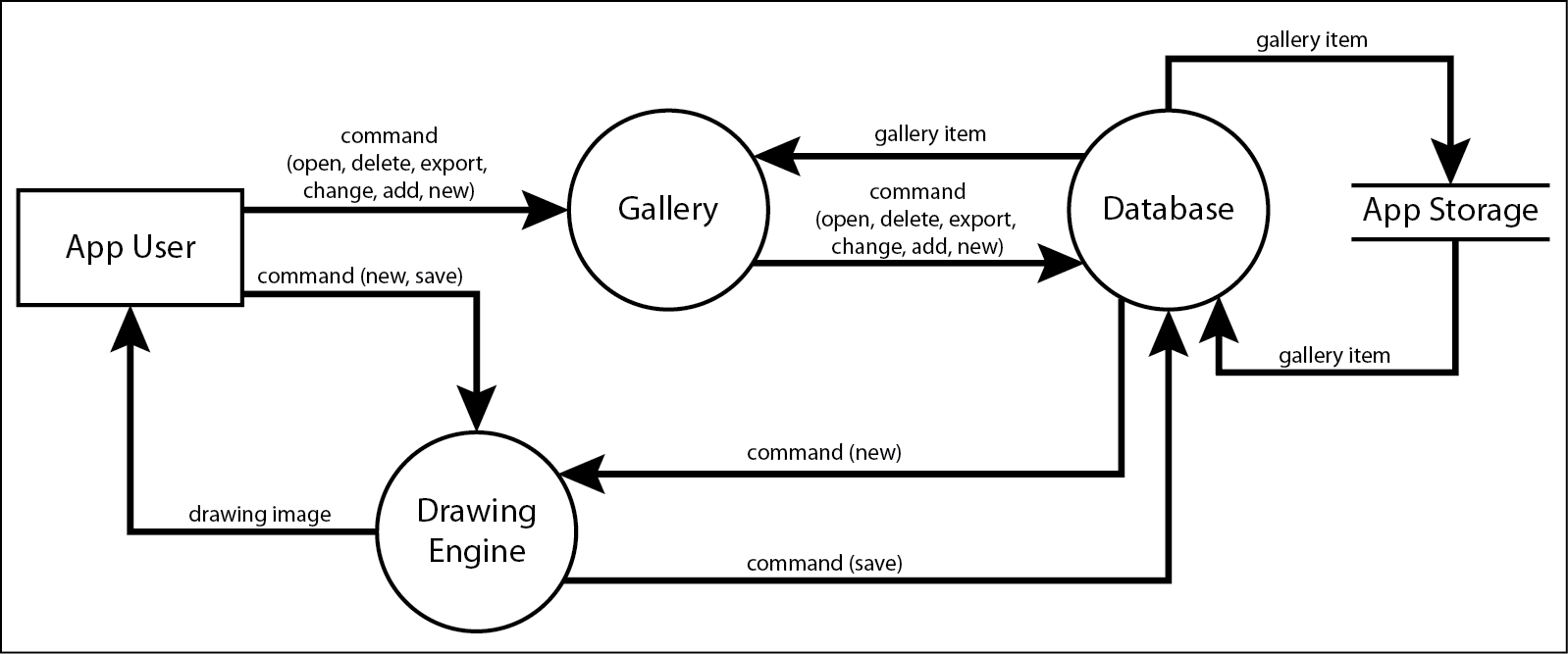


Figure 6 – Command Data Flow Diagram

# Annex C – *Command* Dictionary

The data dictionary below provides a brief explanation of the commands shown in Annex B – *Command* Data Flow Diagrams.

|  |  |
| --- | --- |
| Command | Description |
| *Add* | Adds the specified *gallery item* to the *gallery*. |
| *Change* | Changes a *drawing object* to a *stack*. |
| *Delete* | Deletes the specified *gallery item.* |
| *Export* | Exports the specified *drawing object*. |
| *New* | Creates the specified *gallery item.* |
| *Open* | Opens the specified *gallery item*. |
| *Save* | Saves the specified *gallery item.* |