# Trace2Win

*Robert Li, Seunghoon Park, Hansong Peng, Angela Wu*

# User Manual

## Main Menu

When you start the app, you begin at the main menu screen (fig. 1). Here you will find a navigational menu where you can create and practice characters, words, and lessons, as well as import and export sets of lessons.

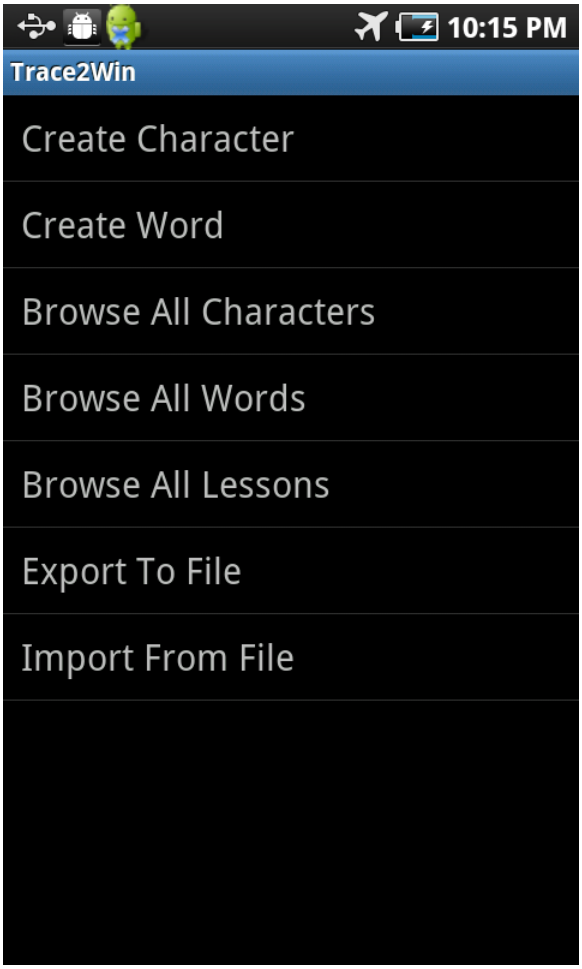


Fig. 1 MainMenuActivity

## Creating a Character

Let's start by making your first character! Select "Create Character" to be taken to the creation screen (fig. 2). Here, you can draw your character in the dark gray box. The app will remember your stroke order, so be careful to draw it correctly! If you need to start over, press the clear button. When you're done, press the save button to save your character.

After you save the character, you'll be taken to the ID and tag screen (fig. 3), where you can add IDs (key-value pairs) and tags to identify and classify your new character. Press the save button to go back to the main menu when you're done.

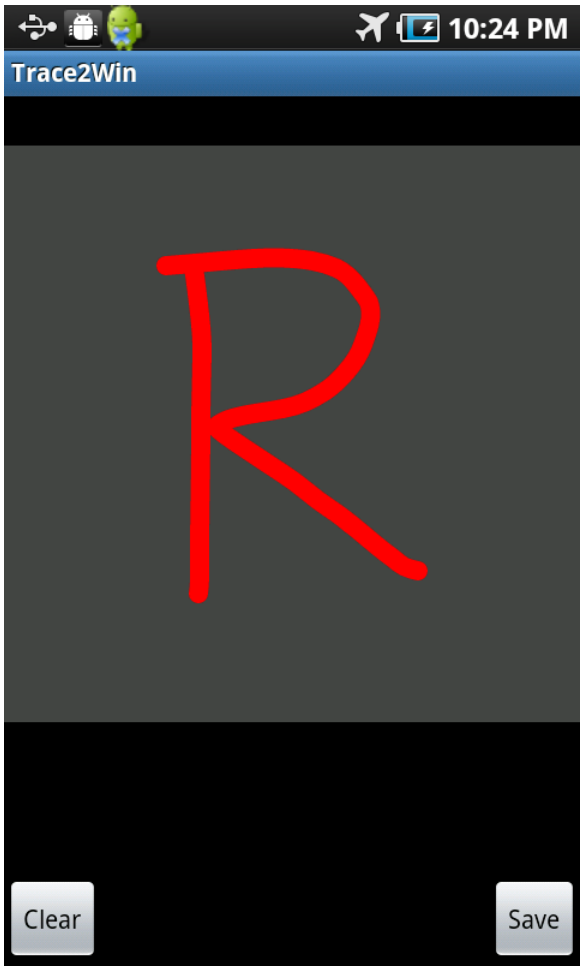

Fig. 2 ViewCharacterActivity

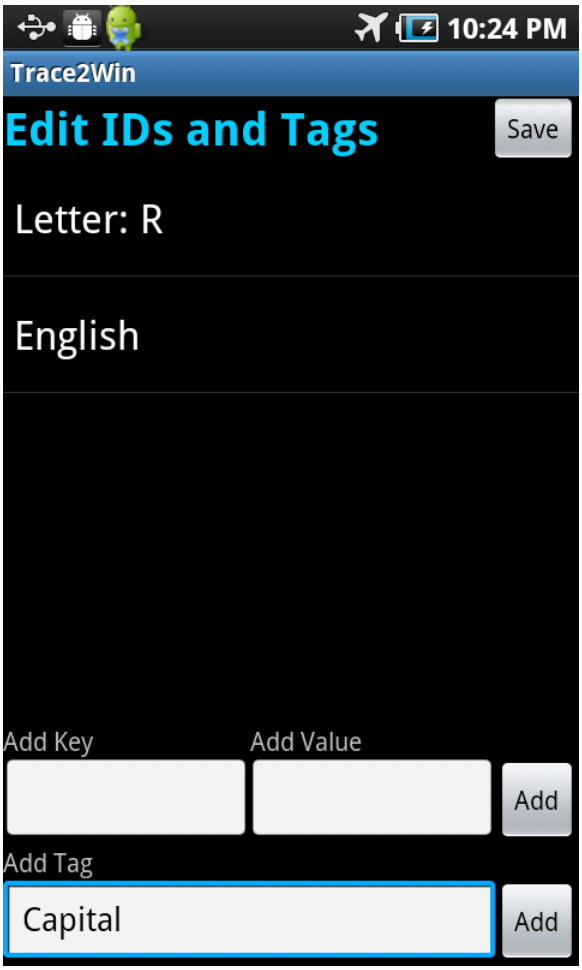

Fig. 3 TagActivity

## Viewing Characters

After creating a character, view it by selecting “Browse All Characters” from the main menu (fig. 4). Clicking on the character will take you to the character view screen (fig. 5), and long pressing on a character will allow you to edit its tags, delete it, or reorder it.

From the character view screen, toggle between playback and practice mode using the buttons in the bottom-right corner. If you need to edit the character, you can return to the character creation screen by pressing the edit button.

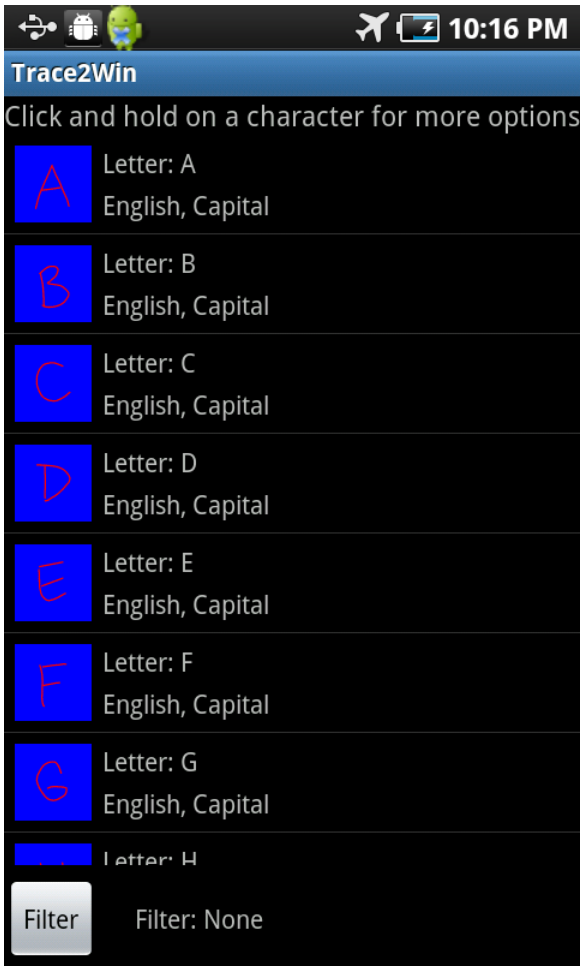


Fig. 4 BrowseCharactersActivity

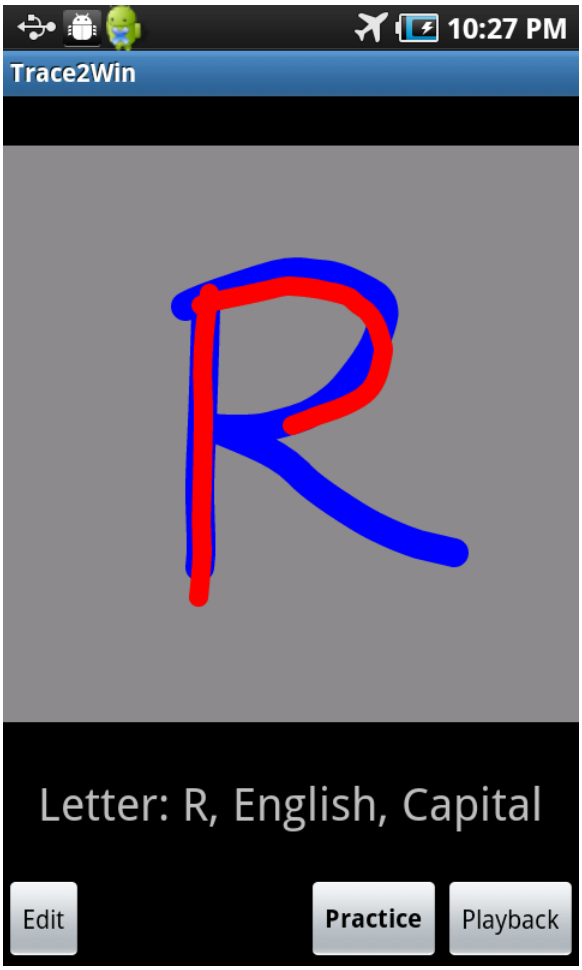


Fig. 5 ViewCharacterActivity

## Creating a Word

Let's create a word by selecting "Create Words" from the main menu—this takes you to the word creation screen (fig. 6). Create your word by selecting the characters it's composed of, one by one. When you're done, click the save word button, and save the word to a lesson. Then, you can click the add tags button to add IDs and tags to the word (fig. 7). This is the same process as adding it to characters.

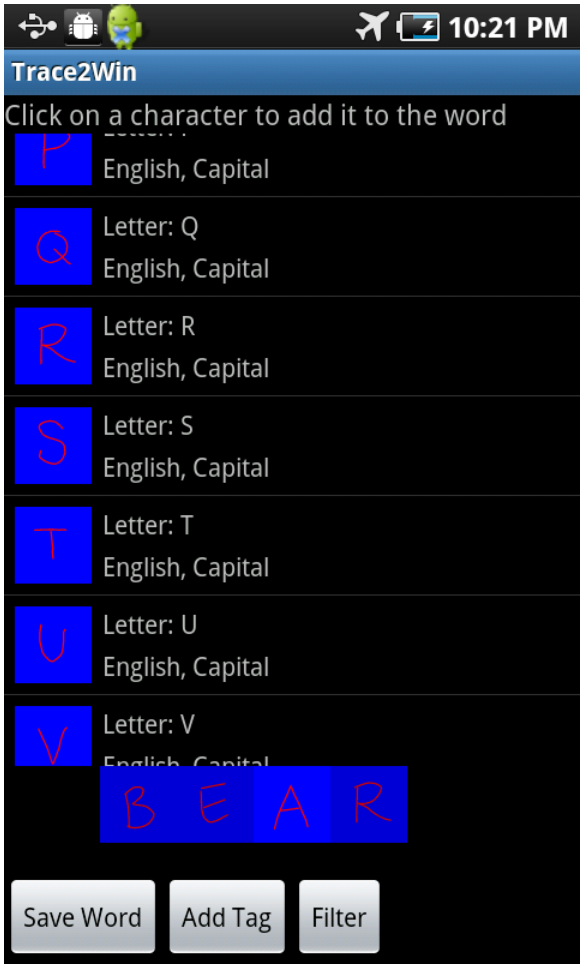


Fig. 6 CreateWordActivity

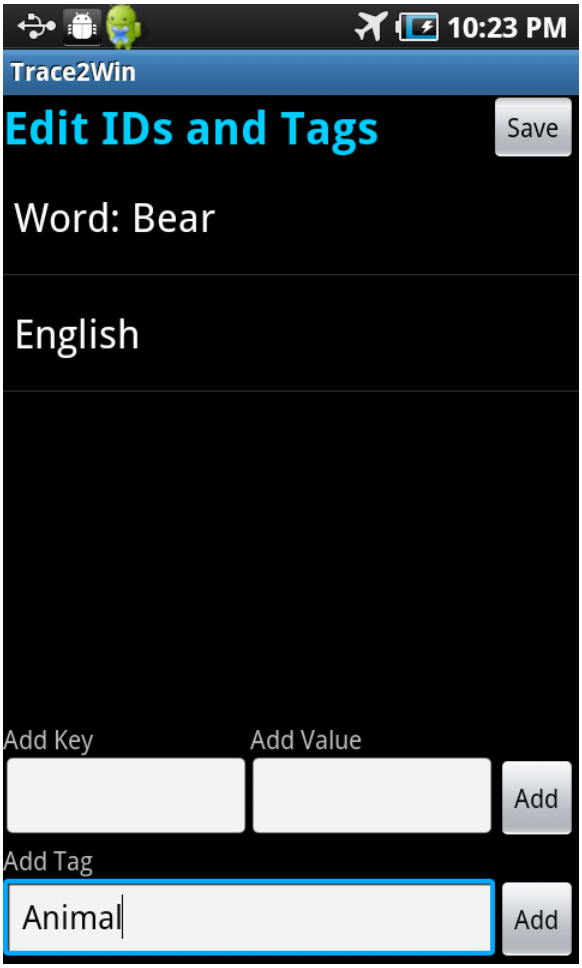


Fig. 7 TagActivity

## Viewing Words

To view words, select “Browse All Words” from the main menu (fig. 8). Clicking on the word will take you to the word view screen (fig. 9), and long pressing on a word will allow you to add it to a lesson, edit its tags, delete it, or reorder it.

From the word view screen, toggle between playback and practice mode using the buttons in the bottom-left corner. In practice mode, you will automatically progress to the next character after you finish tracing the current one. To view a particular character in the word, click on that character in the word at the top of the screen.

Fig. 8 BrowseWordsActivity

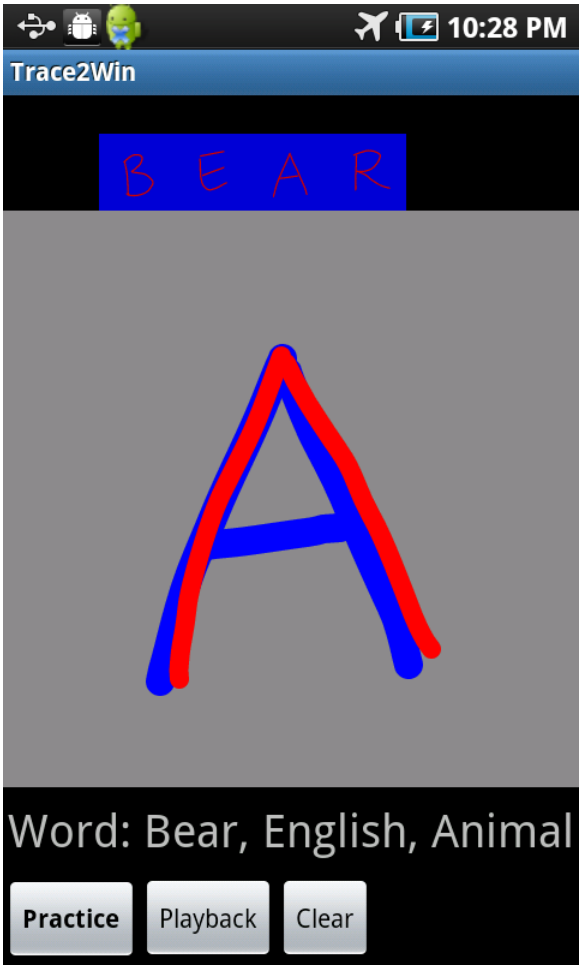

Fig. 9 PhrasePracticeActivity

## Viewing Lessons

All of your lessons can be viewed by selecting “Browse All Lessons” from the main menu. This takes you to the browse lessons screen (fig. 10), where you can select an individual lesson and view the words it contains. When practicing a lesson, you will automatically progress to the next word once you finish tracing the current one.

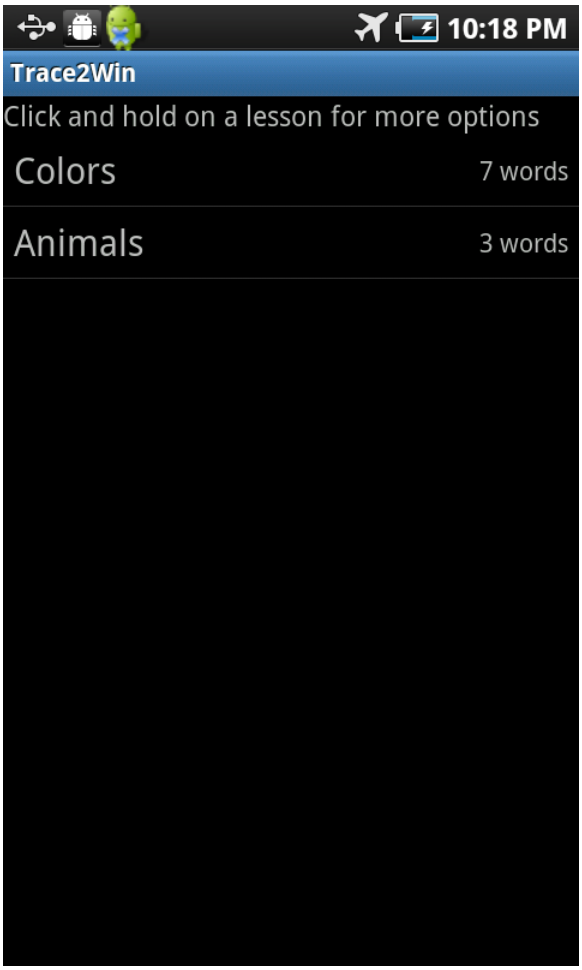


Fig. 10 BrowseLessonsActivity

## Import and Export

Trace2Win is considered an admin app, and as such, is the only way to create characters, words, and lessons. To distribute these to a user, we will export to a file on the SD card. Begin by selecting "Export To File" from the main menu, which brings up the export screen (fig. 11). Select the items you would like to export. Note that "All Characters" will export every character that's currently on the app. When you're done, click the export button to create a new bundle. Note that the file extension ".ttw" will be added to the name of the bundle.

To import a TTW file, select "Import From File" from the main menu. This brings you to the import screen (fig. 12). Select the bundle you'd like to import, and the app will import the characters, words, and lessons. Every item will be added, and none of your current items will be lost. However, if the import contains an item that you already have, it will be ignored, i.e. no duplicate item will be created.

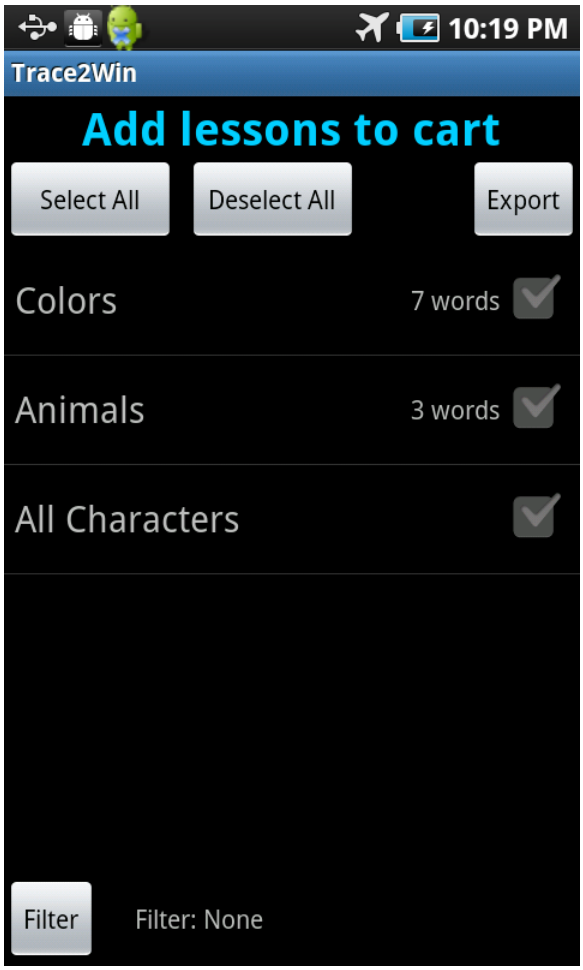

Fig. 11 ShoppingCartActivity

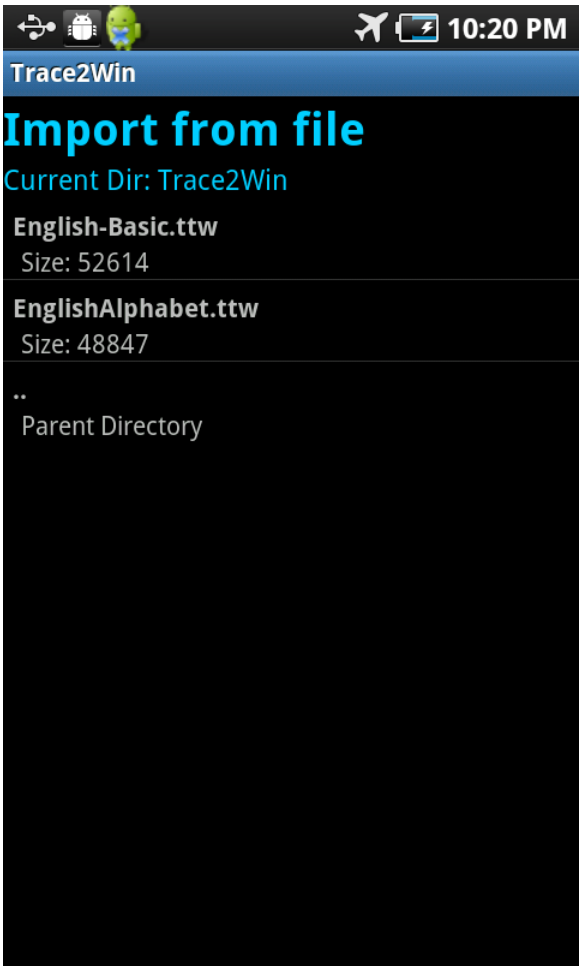

Fig. 12 FilePickerActivity

## TraceThis

As mentioned above, Trace2Win is an admin app. The user app, called TraceThis, is a stripped down version of Trace2Win, and simply has items removed from the main menu (fig. 13). Each item functions the exact same way as the admin app.

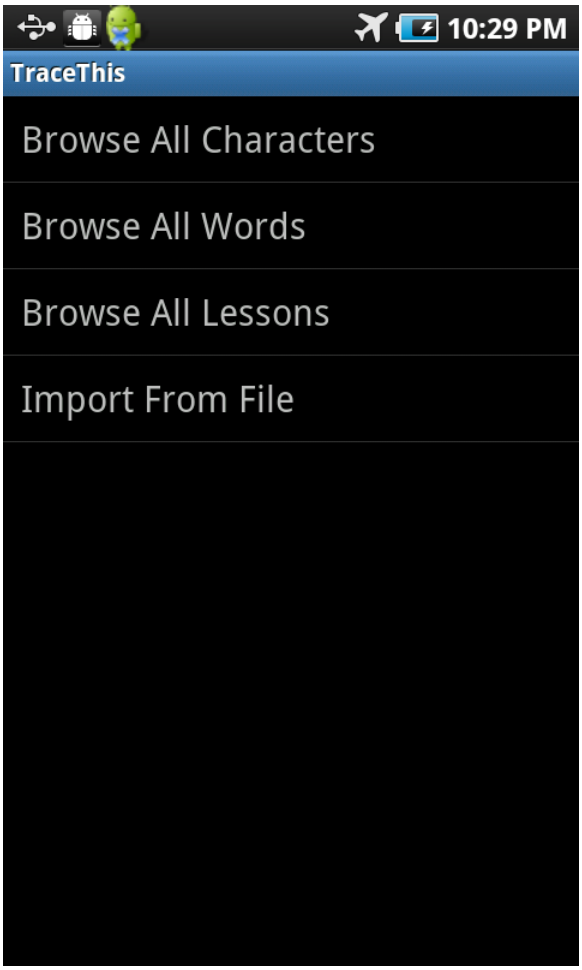


Fig. 13 TraceThis.MainMenuActivity

# Technical Documentation

## Fig. 1 MainMenuActivity

It displays available menus as a ListView. The activities are implemented in the "TraceLibrary", they are ViewChatacterActivity, CreateWordActivity, BrowseCharactersActivity, BrowseWordsActivity, BrowseLessonsActivity, ShoppingCartActivity, FilePickerActivity.

## Fig. 2, Fig. 5 ViewCharacterActivity

Helper Class: DbAdapter
This activity is used when creating a character and displaying a character to the user.
When creating, the activity displays a CharacterCreationPane.
When displaying, there are two different modes: Practice, which is handled by CharacterTracePane, and Playback, which is handled by CharacterPlaybackPane.
When the user selects the save button, the activity sends the LessonCharacter to the database to be saved, and launches the TagActivity and passing it the character's id.

## Fig. 3, Fig. 7 TagActivity

Helper Class: DbAdapter
This activity displays key-value pairs and tags of the selected LessonItem which has three different types: CHARACTER, WORD, and LESSON. The user can reorder/delete key-value pairs and tags by long pressing on one of them, and can also add new ones. When editing, onContextItemSelected() will be called. When entering a new tag and click 'Add', onAddTagButtonClick() is called, grabbing the tag inputted into the EditText and then calling on DbAdapter.createCharTags/createWordTags depending on what type the Intent passed in. The same goes for adding a key-value pair, except that the DbAdapter.createKeyValue is called instead.

## Fig. 4 BrowseCharactersActivity

Helper class: DbAdapter, LessonItemListAdapter
This activity displays saved characters and has a filtering feature. The characters are arranged in a ListView using LessonItemListAdapter. When the user clicks on a character, the id of the character is passed in the Intent and also that the mode is "Practice". This is through the clickOnItem() method that is called within onListItemClick(). The user can long pressing a particular character, which calls onCreateContextMenu(), to edit tags/reorder/delete. If the user wants to add tags, then it launches TagActivity with the character ID and the type "CHARACTER" passed in the intent. If the user wants to delete the character, then the DbAdapter calls on deleteCharacter(id), where the id is the character id. If the user wants to filter the characters by some key words, click "filter" button and it will launch onClickFilter(). The showToast() method is called after either context menu item is selected, and the showToast() method simply displays the message of whether what was selected was successful or not.

## Fig. 6 CreateWordActivity

Helper class: DbAdapter, LessonItemListAdapter
Test: CreateWordActivityTest
This activity displays saved characters and lets the user select them to create a word. A ListView of

all characters in the database are shown, with a filter feature. Selecting a character adds a bitmap to a gallery at the bottom of the screen. Clicking “Save Word” calls on the DbAdapter variable to add the word represented by the gallery into the database. Then a popup shows up. Clicking “Add Tag” checks if the word was saved (there is a saved boolean variable) before it calls on TagActivity to launch.

### Fig. 8 BrowseWordsActivity

Helper class: DbAdapter, LessonItemListAdapter
This activity displays saved words and has a filtering feature. It takes in the id of a lesson in which it will display. If no id is supplied (the case when you enter this activity from the main screen), it will display all words created.
Most methods of this activity work like the ones of BrowseCharactersActivity.

### Fig. 9 PhrasePracticeActivity

This activity handles practicing and playing back a phrase of word. The list of characters at the top of the page is a gallery of bitmaps of the characters in the phrase. These bitmaps are created using the BitmapFactory class. The page uses a ViewAnimator to track all of the character panes, when the user changes between modes, the activity swaps between CharacterPlaybackPane and CharacterTracePane being stored in the ViewAnimator (this is done in a somewhat hacky fashion, and can be done more elegantly). There is a handler for auto-progressing when practicing characters in a selected word. Most of the rest of the functionality behaves the same as the CharacterActivity (and some of the code is taken from there).

### Fig. 10 BrowseLessonsActivity

Helper class: DbAdapter, LessonListAdapter
This activity displays saved lessons. The Lesson objects are retrieved by the DbAdapter.getLessonById after grabbing all of the lesson id’s in the database. These Lesson objects are placed in LessonListAdapter. When the user clicks on a ListView item, the BrowseWordsActivity is launched to see the phrases for the lesson selected. The selected Lesson object’s id is passed into an Intent object and the BrowseWordsActivity is launched.

### Fig. 11 ShoppingCartActivity

Test: ShoppingCartActivityTest
This activity handles exporting lessons/characters and has a filtering feature. The method getType() is used to Initializes the type field based on the bundle passed with it, then populates the source and display lists. There are also handlers for select/deselect/filter/export buttons. The method export() also generates the XML string and calls writeStringToFile() to write the string to a file on SD card.

### Fig. 12 FilePickerActivity

Helper class: FileArrayAdapter
This activity handles importing from existing “.ttw” files. The method importFromFile() will import words and lessons from the given file.

### Fig. 13 TraceThis.MainMenuActivity

It displays available menus as a ListView. The activities are implemented in the "TraceLibrary", they are BrowseCharactersActivity, BrowseWordsActivity, BrowseLessonsActivity, FilePickerActivity.

## Shared Helper Classes

### DbAdapter (Test class: DbAdapterTest)

This class creates the database tables, and provides methods to update or query the database. All direct interactions with the SQL occur in this class.

### LessonItem (Test classes: LessonItemTest, LessonCharacterTest, LessonWordTest, LessonTest)

The superclass for LessonCharacter, LessonWord, and Lesson. These classes are object representations of entries in the database, and all information pulled from the database is stored in and accessed from these objects.

### Stroke (Test class: StrokeTest)

Another object representation of a type of entry in the database. Stroke represents character strokes, and is mostly used by the LessonCharacter class.

### Parser

Singleton class for the DocumentBuilder object, so that a new one doesn't have to be made every time XML needs to be parsed.

### LessonItemListAdapter

The list layout for characters or words. It displays bitmap(s) on the left and private/public tags on the right.

## Database Schema

### Character Table

```
    private static final String DATABASE_CREATE_CHAR =
             "CREATE TABLE Character (_id TEXT PRIMARY KEY," +
             "sort INTEGER);";
```

The Character table holds two variables: the unique "_id", a string consisting of the serial number of the device the character was created on and the timestamp when it was made, and "sort", the position of the character when it is displayed in a ListView with other characters.

### CharacterTag Table

```
    private static final String DATABASE_CREATE_CHARTAG =
            "CREATE TABLE CharacterTag (_id TEXT, " +
```

```
            "tag TEXT NOT NULL, " +
            "sort INTEGER, " +
            "FOREIGN KEY(_id) REFERENCES Character(_id));";
```

The CharacterTag table has three fields. “_id” is the id of the character (from the Character table) that the tag belongs to. “Tag” is the text of the tag itself, and “sort” denotes the position of the tag when displayed among other character tags.

## CharKeyValues Table

```
    private static final String DATABASE_CREATE_CHARKEYVALUES =
            "CREATE TABLE CharKeyValues (_id TEXT, " +
            "key TEXT NOT NULL, " +
            "value TEXT NOT NULL, " +
            "sort INTEGER, " +
            "PRIMARY KEY (_id, key), " +
            "FOREIGN KEY(_id) REFERENCES Character(_id));";
```

The CharKeyValues table contains the data for the keyValue pairs, also known as IDs, of characters. The _id field corresponds to the id of the character the key value pair belongs to. The key is obviously the key, such as “UTF”, or “Pinyin”, and the value would be “U+006A” or “hao3”

## CharacterDetails Table

```
    private static final String DATABASE_CREATE_CHAR_DETAILS =
            "CREATE TABLE CharacterDetails (_id TEXT PRIMARY KEY, " +
            "CharId TEXT, " +
            "Stroke INTEGER NOT NULL, " +
            "PointX DOUBLE NOT NULL, " +
            "PointY DOUBLE NOT NULL," +
            "OrderPoint INTEGER NOT NULL, " +
            "FOREIGN KEY(CharId) REFERENCES Character(_id));";
```

Each entry of the CharacterDetails table is a single point of a stroke of a character. “_id” is primary key of the point, made from the serial number of the device and the timestamp. “CharId” is the character id of the character the point belongs to. “Stroke” is the number of the stroke the point belongs to. “PointX” is the X-coordinate of the point. “PointY” is the y-coordinate of the point. “OrderPoint” denotes the ordering of the point in the stroke; a point with OrderPoint 4 would be displayed right before the point with OrderPoint 5.

## Words Table

```
    private static final String DATABASE_CREATE_WORDS =
             "CREATE TABLE Words (_id TEXT PRIMARY KEY," +
             "sort INTEGER);";
```

“_id” is the unique id of the word. “Sort” is the position of the word when it is displayed in a ListView with other words.

## WordDetails Table

```
    private static final String DATABASE_CREATE_WORDS_DETAILS =
            "CREATE TABLE WordsDetails (_id TEXT," +
            "CharId TEXT," +
            "WordOrder INTEGER NOT NULL," +
```

```
            "FlagUserCreated INTEGER," +
            "FOREIGN KEY(CharId) REFERENCES Character(_id)," +
            "FOREIGN KEY(_id) REFERENCES Words(_id));";
```

"_id" is the unique id of the word the wordDetail belongs to. "CharId" reference a the unique id of a character in the word, and "WordOrder" denotes the position of that character within the word. "FlagUserCreated" used to refer to whether the word was autogenerated as a result of creating a character. However, this functionality has been removed, but the code has not yet been updated to reflect this change in functionality.

## WordsTag Table

```
    private static final String DATABASE_CREATE_WORDSTAG =
            "CREATE TABLE WordsTag (_id TEXT, " +
            "tag TEXT NOT NULL, " +
            "sort INTEGER, " +
            "FOREIGN KEY(_id) REFERENCES Words(_id));";
```

"_id" refers to the unique id of the word the tag belongs to. "Tag" is the text of the tag. "sort" is the ordering of the tag in when the tag is displayed with other word tags.

## WordKeyValues Table

```
    private static final String DATABASE_CREATE_WORDKEYVALUES =
            "CREATE TABLE WordKeyValues (_id TEXT, " +
            "key TEXT NOT NULL, " +
            "value TEXT NOT NULL, " +
            "sort INTEGER, " +
            "PRIMARY KEY (_id, key), " +
            "FOREIGN KEY(_id) REFERENCES Words(_id));";
```

"_id" is the unique id of the word the keyValue pair belongs to. "key" and "value" are the texts of the key and value. Sort is the position of the keyValue pair relative to other keyValues of words.

## Lessons Table

```
    private static final String DATABASE_CREATE_LESSONS=
            "CREATE TABLE Lessons (_id TEXT PRIMARY KEY,"+
            "name TEXT, " +
            "sort INTEGER);";
```

"_id" is the unique id of the Lesson. "Name" is the name of the lesson. "Sort" is the position of the word when it is displayed in a ListView with other words.

## LessonsDetails Table

```
    private static final String DATABASE_CREATE_LESSONS_DETAILS =
            "CREATE TABLE LessonsDetails (" +
            "LessonId TEXT, " +
            "WordId TEXT," +
            "LessonOrder INTEGER NOT NULL, " +
```

```
            "FOREIGN KEY(LessonId) REFERENCES Lessons(_id)," +
            "FOREIGN KEY(WordId) REFERENCES Words(_id));";
```

This table contains information on which words go in which lessons. “Lessonid” corresponds to the unique id of the lesson the word belongs to. “WordId” is the unique id of the word. “LessonOrder” is its display order within the lesson.

## LessonTag Table

```
    private static final String DATABASE_CREATE_LESSONTAG =
            "CREATE TABLE LessonTag (_id TEXT, " +
            "tag TEXT NOT NULL, " +
            "sort INTEGER, " +
            "FOREIGN KEY(_id) REFERENCES Lessons(_id));";
```

“_id” refers to the unique id of the lesson the tag belongs to. “Tag” is the text of the tag. “sort” is the ordering of the tag in when the tag in question is displayed with other lesson tags.

## XML Format

```
<ttw name="English Colors">      the “name” attribute denotes the name of the file

First, the information for the characters included in the export, with each character
surrounded by <character></character> tags


    <character id="1234">    id= unique id of the character
        <tag tag="English" /> tag = tag text
        <id key="Letter" value="A" /> id elements contains key value pairs for the
character
        <stroke position="0"> This is followed by stroke information, with position =
order of the stroke
            <point position="0" x="5" y="5" /> as the schema, position = orderPoint,
x= PointX, y = PointY
            <point position="1" x="6" y="6" />
        </stroke>
        <stroke position="1">
            <point position="0" x="5" y="5" />
            <point position="1" x="6" y="6" />
        </stroke>
    </character>
    <character id="1235">
        <tag tag="English" />
        <id key="Letter" value="B" />
        <stroke position="0">
            <point position="0" x="5" y="5" />
            <point position="1" x="6" y="6" />
        </stroke>
        <stroke position="1">
            <point position="0" x="5" y="5" />
            <point position="1" x="6" y="6" />
        </stroke>
    </character>
```

```
After the character information, the information about lessons and their words
    <lesson id="666" name="Colors">    the id = lessonId, and name is the lesson name
        <word id="900" position="0">   id = id of the word, and position = position
within lesson
            <tag tag="English" />   These are wordtags that correspond to the word
            <tag tag="Color" />
            <id key="Word" value="Red" />   key values of the word
            <character id="1234" position="0" />  characters in the word, all
characters must be part of the earlier character section of the xml. id = id of the
character, position = position of character within the word
            <character id="1234" position="1" />
        </word>
        <word id="901" position="1">
            <tag tag="English" />
            <tag tag="Color" />
            <id key="Word" value="Blue" />
            <character id="1235" position="0" />
            <character id="1234" position="1" />
        </word>
    </lesson> If more than one lesson is exported, follow with another
<lesson></lesson> bracket
</ttw>
```