Jar of Joy

Alisa Nguyen & Joy Ming
CS50 Final Project, Fall 2011

DESIGN

0. Learning

0a. Acknowledgements

As the learning curve for two new languages was extremely steep, we could not have done it without help from so many individuals, specifically Edwin Guarin, Joe Healy and Nathan Leiby from Microsoft[1], as well as our TF, Joseph Ong[2].

0b. C# and XAML

Much of the time was spent toying with the functionality of Visual Studio and its usage in relation to the Windows Phone 7.1 (WP7 codenamed Mango). We also worked with tutorials and reference materials provided by Edwin and Joe.

0c. Silverlight and XNA

The distinction between Silverlight (used for application pages, i.e. "home", "random" and "info") and XNA (used for game pages, namely "create") created the need to learn not only two languages but also two frameworks. For example, displaying text on XNA involves the need for a spriteBatch to represent an object containing a string as opposed to the simple textblock in Silverlight. Also, passing information between the two was facilitated at first through the use of text/xml files and later on more directly through the declaration of global variables in the app.xaml. We declared a List (called App.Apple.quoteList) in app.xaml to hold all the quotes, and decided to use a globally accessible List so that the user can add their own quotes to this list through the "create" feature and access their own quotes as well as the original quotes through the "random" feature.

1. Data Input/Output

1a. Creating online files

After careful explanation of access to the cloud, all external text and xml files used on the application pages were stored at http://cloud.cs50.net/~jming/ with the proper permissions.

1b. Reading online files using Silverlight

---

[1] Edwin.Guarin@Microsoft.com, Joe.Healy@Microsoft.com, NathanLeiby@gmail.com
[2] jong@cs50.net

When first starting this project, many possibilities for storing the quotes for the "random" page came to mind, including querying to a SQL server, using a WCF request, or using an HTTPGetRequest. After consulting with Joe and toying with code samples online and various examples, we decided to store the quotes as a text file on the cloud and download the text from that file as a string through a WebClient and HTTP GetRequest.

1c. Reading files using XNA

Because XNA does not support StreamReader or many of the File I/O functions for text files that were tried based on Google search results, we had to read the files from a XML file as opposed to a text file. This was done using an XmlReader and parsing the returned text within the tags as a string.

1d. Writing to source file

Another problem we ran into was writing to a globally accessible file using the XNA framework. Although we did extensive research and tried to implement writing to a file on the cloud and writing to a file stored locally (through Isolated Storage), we found a simper option and decided instead to pass strings through a global variable defined in app.xaml.

2. "random" feature

2a. Linking to & reading the text file

The quotes were stored at http://cloud.cs50.net/~jming/quotes.txt and fetched using an HTTPGetRequest. Our application also includes an offline mode that activates automatically if the text file could not be retrieved from the cloud (i.e. if the cloud server crashed). The offline mode reads quotes from a local quotes.txt file. After reading a string from the text file, our application parses through and puts each quote into a List, which we chose to use instead of an array because of its modifiable size and ease in adding/removing elements. The user can add quotes to this source file by using the "create" feature and tapping save, which will add the user-created quote to the global variable.

2b. Shake Gesture

A built-in feature of the WP7 is the ability to sense changes in motion. For this feature we utilized the Windows Phone Shake Gestures library created by Microsoft[3] that senses how many shake motions the user made through the built-in accelerometer. If the user shakes the phone more than four times, a quote is randomly generated (by calling on a random index of the list).

3. "create" feature

3a. Reading text file

We stored the words to be put onto "magnetic poetry" tiles into a XML file and used an XmlReader to access them because the simple reading of a text file is not supported on XNA. In addition, because accessing the server is difficult with XNA, we decided to store the text data for the files in a local XML file.

3b. Creating Tiles

[3] http://windowsteamblog.com/windows_phone/b/wpdev/archive/2011/02/22/windows-phone-shake-gestures-library.aspx

(i) *Array of Tiles*
In order to keep track of all the properties of each element, we created a Tile class that would hold the word, position(x, y) and other aspects of the state of each tile. We created two global static arrays of Tiles that would be accessible throughout the XNA page, one with the original properties of the Tiles before moving and modifying the page(original[]), and one with the current real time properties (Piece[]).

(ii) *Looping to generate the Tiles*
As the creation of the Tiles was a repetitive process, we created a loop that would go through each word on the XML file. Each of the Tiles in the Piece[] array was assigned a word, appropriate state properties and a position based on its index within the array (sorted alphabetically).

(iv) *Tile arrangement*
As the Tiles are arranged in pages, first, and then rows and columns, a loop was made to section the tiles into groups of 18. Then within each group of 18, we used the modular operator on the index to group them into three columns. Each group of three was then placed in one of the six rows.

(v) *Switching pages*
In order to flip between each page of tiles, a spriteBatch was created for the forward arrow, another for the back arrow, and also another to display the current page. We needed to use spriteBatches to represent all our objects (even text-only objects) because that seemed to be the only option supported on XNA. The forward arrow is only drawn if the current page was not the last page and the backward arrow is only drawn if the current page was not the first. The display of the current page is a spriteBatch that updates based on a variable that increases if the forward arrow is pressed and decreases if the backward arrow is pressed.

3c. Interacting with Tiles

(i) *Tapping the Tile*
The TouchCollection state of the TouchPanel is retrieved everytime OnUpdate() is called, and returns the locations of the current touch points. A conditional loop checks to see if a Tile was touched during the current round and if it was, the Tile touched is selected and movement of that Tile is enabled. A variable representing whether or not the tile is touched called "isClicked" was included as a part of the Tile class to keep track of which tile was being clicked at a point in time. The currently clicked Tile will display red text instead of black text. One bug we ran into was the clicking of multiple Tiles accidentally. This was fixed by ensuring the user only clicks on one Tile at a time through a global Boolean variable called "currTile". Also, to ensure that only tiles of the current page were being clicked, onUpdate() only loops through Tiles displayed on the current page.

(ii) *Moving Tiles*
Our original plan was to enable the dragging of Tiles, but because of limited time was unable to include that feature in this version of Jar of JOY. Instead, users can move Tiles by first tapping on the desired Tile, which will change its state to "isClicked" and its text to a red color, and then tapping a second area on the screen that they would like to move the Tile to. To compose a quote, poem, or string, the user must tap the target area and drop the Tile within that area.

(ii) *Creating the target*
A target area, a lavender framed black box, was created on the top portion of the screen through a spriteBatch. When a Tile is dropped into this area, the user can choose to append the word on the Tile to the current composition string by tapping the check mark icon, which also replaces the Tile to its original position. If the user would not like to add the word to their composition, they can tap the x icon, which returns the Tile to its original position. The x icon also deletes the last word appended, and only the last word, which we keep track of through a global variable lastIndex, which contains the index of the last Tile moved.

2c. Creating new quotes

(i) *Current composition display*
A spriteBatch called textBatch was used to create the white bar at the top of the screen and it displays the current string (called textText) the user is working on editing.

(ii) *Appending to composition*
Each Tile contains a field for its displayed word, called Word. When a user drops a Tile to the target area and taps the check mark icon, the Word is appended to the end of the current composition string. In addition, tapping the check mark icon also returns the Tile to its original position.

(iii) *Deleting from composition*
To delete the last word added to their composition, the user can tap the x mark icon. In addition, tapping the x mark icon also returns tiles in the target area to their original position.

(iv) *clear*
If the user would like to reload the create page and delete any quote they may have started to write, a menu can be accessed by tapping the three dots in the bottom right hand corner of the screen with the option called "clear".

(v) *save*
When the user is done creating a quote to be added to the source file, they can tap the three dots in the bottom right hand corner of the screen and then tap the option "save", which will add the current composition string to the List global variable defined in app.xaml. The global variable is accessed through the "random" page, so the next time the user shakes it up, they may see their own quote appear on the screen.

3. Conclusion

3a. Final Acknowledgements

Thanks to all those who were the beta-testers of the application! You helped us find some of our bugs and make the application more intuitive and user-friendly.

3b. Thoughts for the future

We hope you enjoyed learning about and playing with our app, we enjoyed creating it for you ☺
Our plans are to make further improvements (implementing more features, i.e. dragging of tiles) and to upload it to the Windows Phone App Hub sometime in the future.