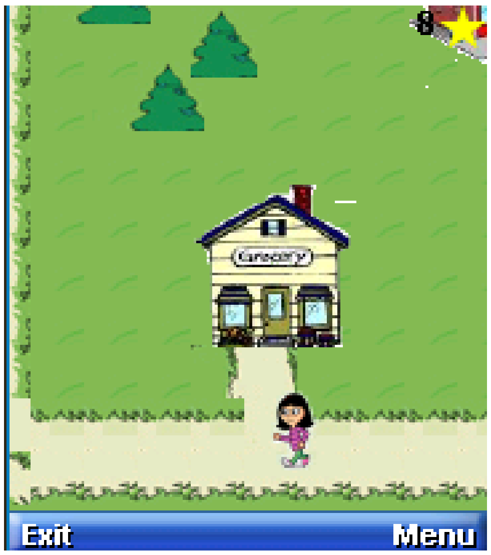
# CSCI 188/297 Final Project Report



## L'il Village

14 December 2009
Cheryl Clopper

### *Summary*

L'il Village is a J2ME application that strives to be an educational game for children. The child chooses a character and is able to navigate around a map to find 30 stars. Some of the village locations are mini games and require the child to answer a set of questions and other locations will earn the child a star upon discovery. At the successful completion of of a mini game, the child will receive five stars.

### *Implemented Features*

A main menu is implemented that triggers an event immediately upon selection. A new game can be started or a previous game can be started if it were saved. Only one game can be saved at any time. A user can choose to be either a little boy or a little girl as they wander through the game, but the Easy/Hard map feature was not fully implemented (only Easy works properly).

The main game screen was developed using a GameCanvas that is larger than the screen to allow for exploration. I used tiled layers and a layer manager to limit where the character was allowed to walk (i.e. roads only) and manage collisions between various sprites to detect whether stars were found or whether games were entered.

The five mini games are:

1) Grocery store: Shape Game – Display a food and then give a set of choices for shape (orange- circle, cracker – square, carrot – triangle, loaf of bread – oval, block of cheese - rectangle)
2) Auto Shop: Color Game – Display a car, then give a set of color choices (red, yellow, blue, orange, green cars)
3) Police Station: Matching Game – Belongings must be matched to the owner who has lost them (police man – hand cuffs, fireman – fireman's hat, child w/ leash – missing dog, woman – missing purse)
4) School: Counting Game – An image is displayed and the child has to select how many people are present in the classroom.
5) Pet Store: Identification Game – Child will be presented with five animals (cat, dog, mouse, bird, fish) and must identify what the animal is from a list of choices.

Each mini game has its own screen and begins with a help screen of simple instructions to explain the intent of the game. To maintain a consistent interface, the high level API was used for all mini games, the help and about screens, and notification screens used for found stars, successfully completed mini games, and missed questions. When a child does not successfully complete a game, they are exited from the location and placed at the original point on the map to find their way back to the location. The game can be saved (or paused by choosing a menu item) at any time and the settings will be saved.

The mini games could be reused as simple midlets that could stand on their own as a set of games, and the structure could be used to put in additional games. It requires a simple edit of Strings and appropriate image files. The game over listener interface could also easily be used with another game, and of course I have used the settings class in multiple labs and the project now and it is very easy to reuse. The village character could be reused with any given map and it would be very simple to add additional maps to this application or additional characters provided they were the same size. In general it is a fairly extensible application, the biggest danger is that the image resources would become larger as the game were expanded.

## *Encountered Issues*

I ran into several minor difficulties, but the most time consuming are addressed here.

### Maps

I attempted to reuse code from the Aquarium lab to read in map tile settings from a file. It seemed like this should be trivial, but I ran into several issues. It wasn't reading properly, so I ensured that it was formatted exactly as my aquarium files (all on one line and a csv file). I needed to be able to consume numbers that could be more than one number long, so I had to make the code more robust in this respect. That was a fairly simple change, but still didn't immediately solve my issue. I tried reducing the size of my map and minimizing the number of tiles I was bringing in. This is why I implemented the selection of a hard and easy map though I never went back and implemented the hard map fully. I then realized that since I was pushing the tile information into a byte array, that I could have a maximum of 128 tiles, which I was beyond for both maps. I then adjusted to a short array and everything worked beautifully.

**Character Movement**
I decided to do all movement of my village character with a single set of images in a single direction to reduce the amount of space required for all the images since this was a very image intense program. I first tried to use two different sizes of sprites, however, it turned out to be very difficult to get the sprites to behave appropriately as far as where they moved when the direction they were supposed to be moving in changed. I then made them the same size. I also originally tried to implement the sprites so that they were rotating about their feet. This seemed cool, but got me into lots of trouble as far as them getting stuck and was fairly unintuitive to use. I then readjusted so that the sprites moved about their middle, and this was much simpler and intuitive to react to the movements of the sprites.

**Choice Group Images**

In some of the mini games, I wanted to have images that were next to the choices that the user could select. Especially for the police station matching game, I wanted to be able to show each character next to their name. I chose images for all the selections and set them up, but unfortunately, by default, a choice image is very small. If my images were that small, it was not worth having them there. There was no setting I could find that would use the image in the size that it was provided in order to build the choice image. I found this very disappointing, but worked around it by providing names that I thought would make the game easy enough to win.

## *Test Procedures*

The components of the application were tested as they were developed through unit testing. Any portions of the game that could be tested at a given time were exercised. This helped me identify bugs quickly and I had a good idea of where they resided since I tested the code in very small chunks. Although it seemed time consuming, I think it was very effective in preventing larger issues from occurring.

A full “integration” test was also completed on numerous occasions and the following features were exercised:

The application was deployed over the air to ensure that all settings were maintained. All menu options on the main menu (i.e. starting a new game, resuming a game, exiting, or viewing help/instructions) were exercised. One character was chosen and the easy map was selected. Then I played the entire game. I made sure to ‘miss’ some questions to test that functionality and saved and exited before and after doing so. I ensured that the star total was correct and that games that had been successfully played were not allowed to be played again. I also ensured that the current placement of the character was at the same location as when the game was saved. I was successfully able to complete the game and then start a new one. I then was able to win the game and exit. I started the game as the other character and repeated the entire test.