Trie to build it

Project 2: Aesthetic Computing

COP6703 Spring 2012

14th March 12, 2012

Submitted By:

Nitin Agrahara (Frontend Developer and Designer)

Abhiram Jagarlapudi (Backend Developer and Designer)

Meghala Shanmugam (Support Developer)

Trie to build it is an interactive game which aims to implicitly educate the user about the ”Trie” (pronounced as ’try’ )data structure. Most non CS people would have nothing to do with formal data structures apart from throwing a cursory glance once in a while. This project aims to bind something that most people are familiar with but unaware of the data structure scenario involved – Scrabble. A description of tries follows.

Representation of data structure: Trie

A **trie**, or **prefix tree**, is an [ordered tree](http://en.wikipedia.org/wiki/Ordered_tree_data_structure) [data structure](http://en.wikipedia.org/wiki/Data_structure) that is used to store an [associative array](http://en.wikipedia.org/wiki/Associative_array) where the keys are usually [strings](http://en.wikipedia.org/wiki/String_(computer_science)). Unlike a [binary search tree](http://en.wikipedia.org/wiki/Binary_search_tree), no node in the tree stores the key associated with that node; instead, its position in the tree defines the key with which it is associated. Sub trees originating from a node have the same prefix up-to the node defined by the path from the root to the node.

The data associated with each node in the trie is as follows:

* A char representing the letter associated with the node.
* An array of 27 pointers to nodes representing (a-z) characters and a special optional character. The index of a is 0, b is 1 and z 25.
* A Boolean *isEnd* representing the end of the string.
* A static string *foundtree* for the trie instance representing if the string is found by recursion.

The three main methods of the trie are:

*Void insertSuffixAt (char c, int insertedId, int idToInsertAt)*

Inserts a character with the id insertedId at the id given by idtoInsertAt. Also sets the isEnd marker to true. This runs a depth first search algorithm to search for the node to be inserted at and inserts the node at that id.

*int insertInString (String s, int p)*

This directly inserts a string, compressed into a node. Is used for setting the start of the game.

*String dfsSearchLetter (String currStr, int endId)*

This takes input as the last inserted id and returns the word associated with a path from the root to that node.

Additional methods implemented are to print the tree and calculate positions of the nodes of the tree (nicely) on the fly.

Implementation

The logic of the game runs around a modified version of scrabble. It is clear why blocks in a scrabble may be represented by a trie data structure. The whole point is compression. The user has to build the trie from the root up using the alphabets available in such a way that every addition to the trie results in a valid English language word. The user involvement is achieved through a Microsoft Kinect. The alphabets represented by scrabble blocks need to be grabbed using a “grabbing” action and pulled to the point where it is intended to be inserted.

*Trie To Build it* was built in Processing with the help of the following libraries:

SimpleOpenNI – For the Kinect input

Minim – For audio.

The SimpleOpenNI library is additionally dependent on the NITE SensorKinect and Sensor packages which need to be installed. SimpleOpenNI allows us to access the Microsoft Kinect Input through a series of variables and methods. We have accessed the right hand of the total skeletal information available to us in order to display the hand.

The insertion and deletion from trie are achieved not just because of graphical representations; we have implemented a fully functioning TRIE within the project. This enables us to insert/delete into the trie from specific locations based on graphical coordinate input.

We went ahead to implement our own gesture for Kinect. Although the SimpleOpenNI does provide us with three inbuilt gesture recognition modules, none matched our goal of ‘User Intuition’. A user will obviously pick/grab a block in a real life scenario. He will not wave/raise his hand or snap his fingers to obtain a block.

The Game begins with one alphabet already inserted into a trie. This is the base for the trie. The user can see a traditional wooden scrabble stand with wooden blocks on the top right hand. The user will be able to move the hand on the screen using his/her own right hand.

The input available from the Kinect is only 640 x 480 pixels. We have scaled the input in order to obtain a larger play area. Once the user grabs the block he/she wants a message cloud on the bottom right gives the user information about what he can do with the block. The grab action attaches the block to the users hand and thus achieving the drag ability. The insert operation is achieved by holding the block in the hand over the node in the trie for around a second’s time. Whenever a block is moved over the destined node, the node starts sparkling to indicate the insertion process.

Upon insertion, the new block moves to its position in the root. Every time a block is inserted all other roots accordingly change position to accommodate their new cousin.

A csv file containing a list of most English words is loaded in the memory at startup. This object called as the dictionary is used as reference to check if the letters inserted by the user make sense.

Recycle

It is possible that the picks up a block and realizes that it cannot be inserted anywhere in an constructive manner. In such a case, he/she has to drag the block over to the recycle sign on the screen. The instruction cloud informs the user that he needs to keep it there for some time to process the instruction. Upon recycle, the block moves back to its original position in the stand.

Reset

At any point the user can restart the game but hitting the reset button which brings out the thunder cloud which destroys all the blocks in the trie one after the other. Once everything is destroyed, the game restarts again with a new set of alphabet blocks and another trie with a different alphabet as the root.

Target Audience

The game is intended to be used as a teaching tool. As mentioned earlier, most non- computer science people would not be interested in what a data structure is and how we are surrounded by them. This game aims to educate them of how simple things that we see every day can be associated with data structures. By forcing the user to insert characters in a way that they mean something, the game enables the user follow the data structure and the operations on it.

By insertion, the Trie insert function is achieved.

By reset, the trie delete function is achieved.

During insertion, the trie search method is achieved using which the decision of whether a word is found or not is made.

What the user learns

* The representation of a data structure in a day to day game.
* What a trie is and how scrabble can be used to illustrate a trie.
* How an insert, delete and search functions of a trie work.

Shortcomings

Lack of space. Since each node can ideally have 26 children, an case where the user inserts 26 children at a node, the node overlap and become illegible.

Instability in Kinect input. The Kinect depends on a lot of factors such as range, light etc which influence how skeleton tracking is done. A minor deviation in any will cause some amount of randomness in the Kinect skeletal data information which forces the program to display suddenly show the hand at a distorted location for some amount of time.

Demonstration

A short video of our project, giving an explanation and a demonstration of the game can be found at: