*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Feature Document

ID #725 Implement Hash-Table User Interface Scene

**Name: Daniel Khawand**

**Team Member(s): Hamilton Chevez, Bernardo Pla, Pachev Joseph, Daniel Rivero**

**Project: Web VR 1.0**

**Product Owner(s)**: Francisco Ortega

**Mentor(s)**: Francisco Ortega

**Instructor**: Masoud Sadjadi

**User Story Name:**

As a student, I would want a fully functional Hash-Table Interface Scene in WebGL or WebVR so that I could understand Hash-Tables better.

Acceptance Criteria

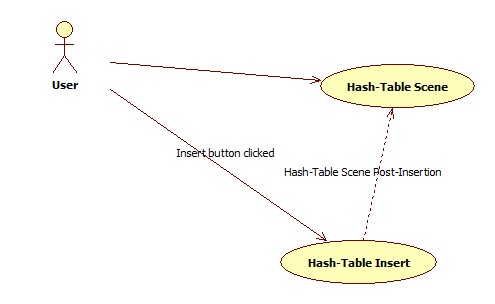
* Takes a size from the user.
* Takes data from the user.
* Data storage visualization included.

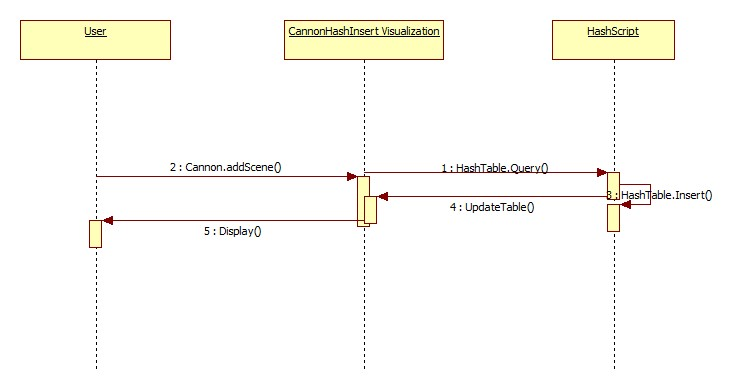
**Use Case**

* Name: Hash-Table Insert PerfectSquares mod 10
* Actor: User
* Preconditions: User must have WebGL enabled and must have clicked the Hash-Table Insert Scene.
* Description:
  + User enters scene
  + User inserts a perfect square automatically each time the

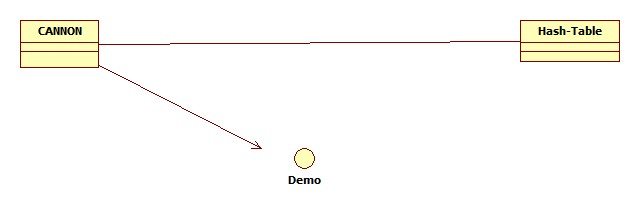
“Hash-Table Insert” Button is clicked.

**Use Case Diagram**

**Sequence Diagram**



**Class Diagram**



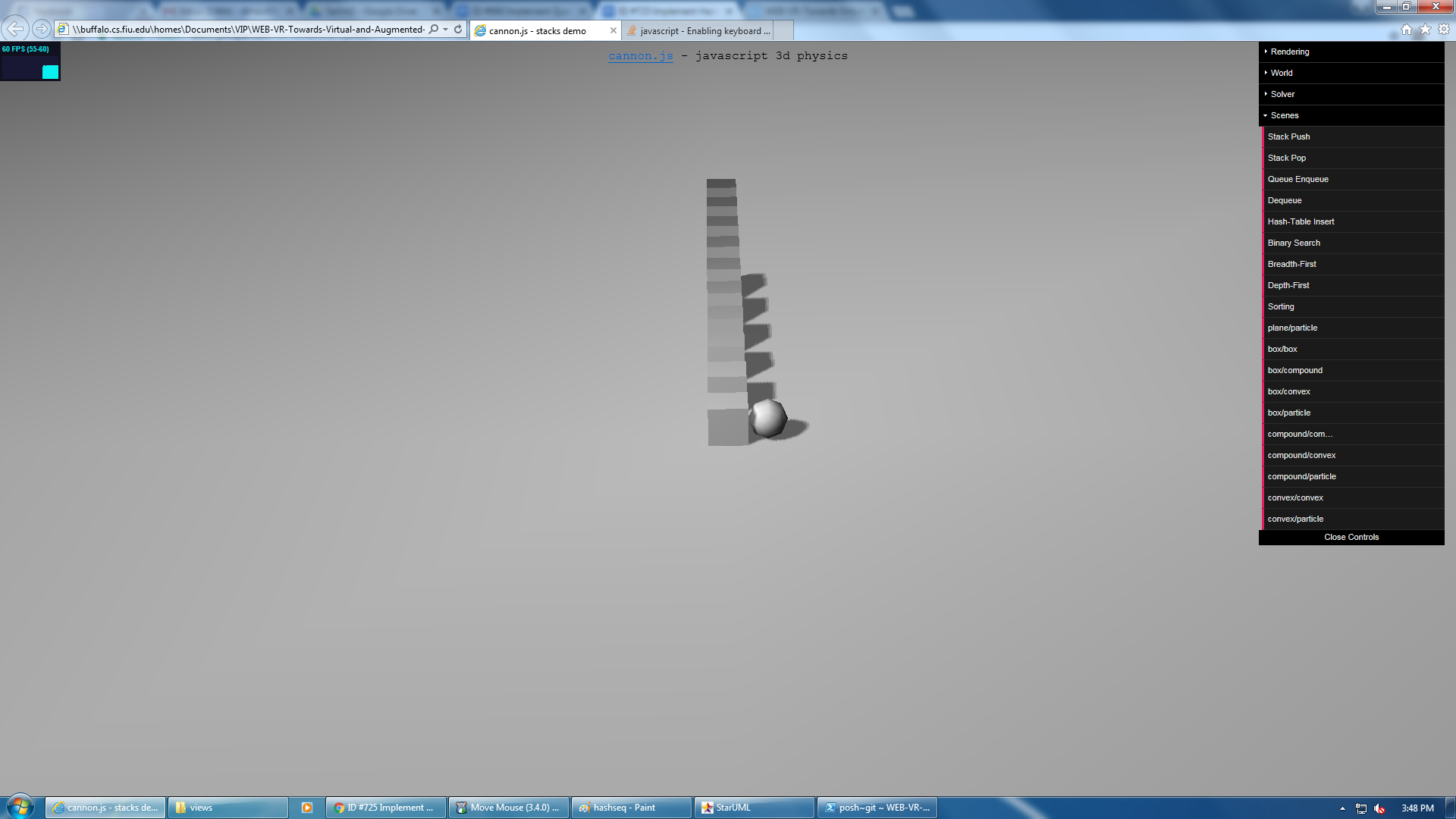
**Unit Test**

* Test case ID: Hash-Table Separate Chaining
* Description/Summary of Test: The “Hash-Table Insert” Function will be tested with an embedded hashing function already implemented and a user input of insertion clicks. h(k) = k mod 10, where k is a perfect square.
* Pre-condition: Pre-defined hash-function, WebGL enabled, and “Hash-Table Insert” scene button clicked.
* Expected Results: Visual of a properly filled separate chained Hash-Table with perfect squares as input values
* Actual Result: Visual of a properly filled separate chained Hash-Table with perfect squares as input values
* Status (Fail/Pass): Pass

**Integration Test**

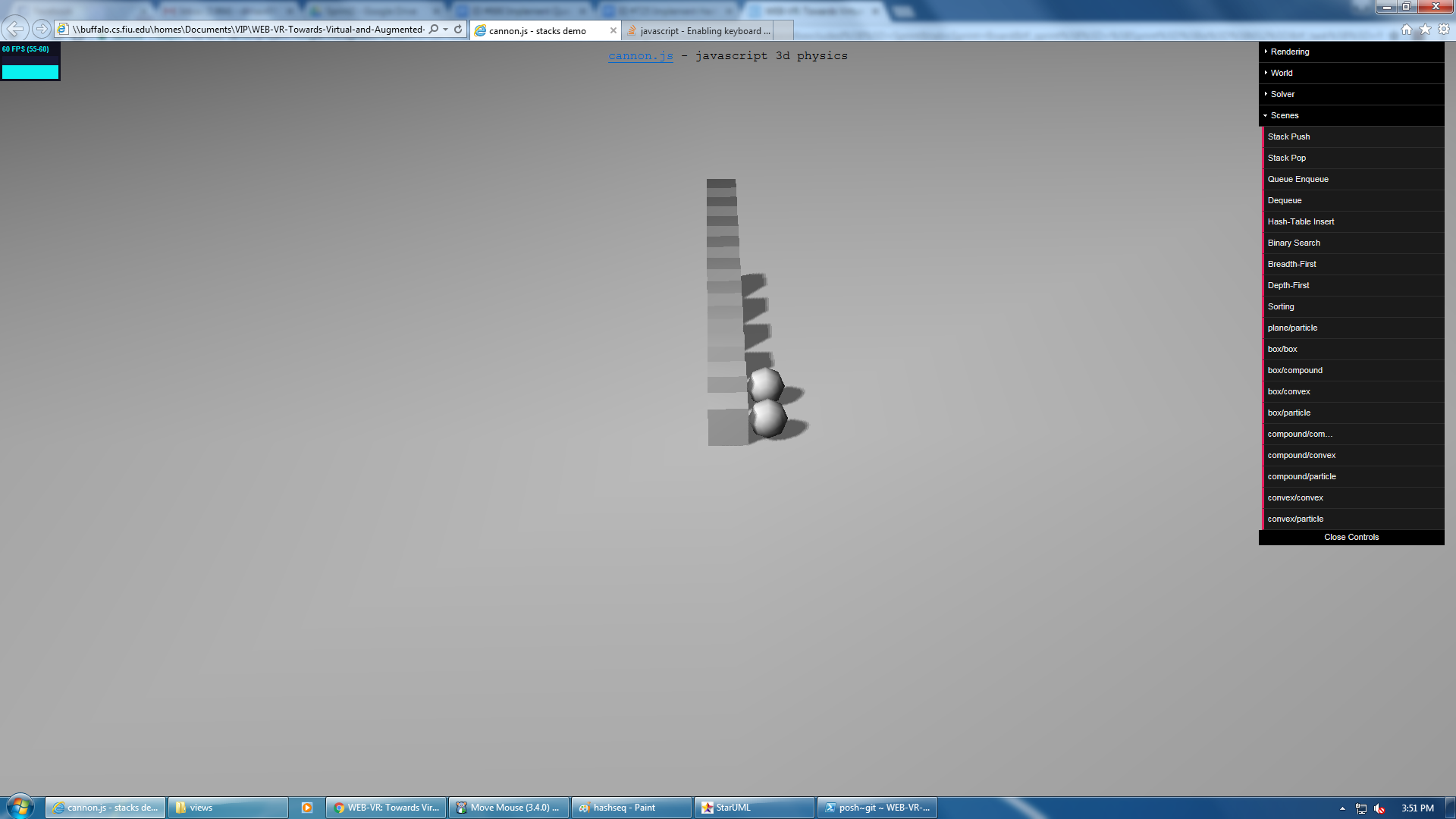
**Visual User Guide**

* Hash Function: K mod 10, where K is a perfect square.
* Table size = 10
* Separate Chaining used, each chain is of size 10.



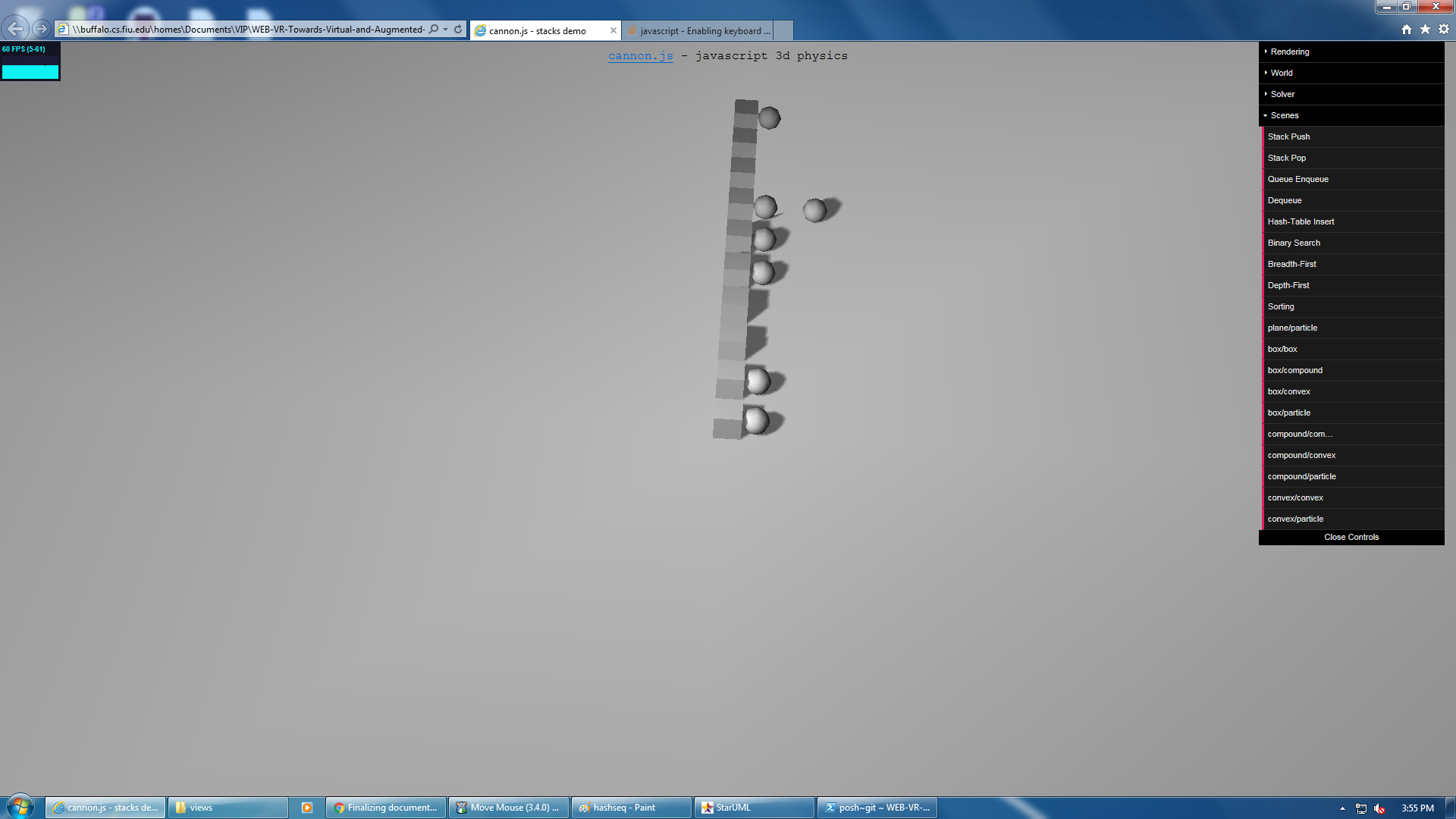
The Scene above is after inserting the number 0. The box with the ball next to it is index 0 of the Hash-Table and the index number goes up from there. The last box is of index 9.

The next key to insert is the number 1. The result is shown below:



Then, the next key will be 4. So a spere will pop up next to the fourth box.

As for collision handling, an example is shown below when the key ‘36’ is inserted where ‘16’ is already at the same Hash Index.



A collision occurred where the two spheres are shown. This visually represents the resulting linked-lists from separate chaining collision handling.