Learning heaps through gameplay

Antoine Campbell, Amanda Chaffin, Bethany Miller, Drew Hicks, Dr. Tiffany Barnes
acampbel@uncc.edu, achaffin@uncc.edu, bm74733@appstate.edu, ahicks37@uncc.edu, tiffany.barnes@uncc.edu

Antoine Campbell

Graduate Advisor:
Amanda Chaffin

Professor Advisor:
Dr. Tiffany Barnes

Introduction

This summer in the Games + Learning Lab, we worked on two separate Game2Learn educational games. Bunny Generals is a game that allows players to visualize and walk through sorting algorithms by placing them in control of the advancement of the algorithm. My contribution to Bunny Generals was design, programming, and assisting with the design and execution of the user study. The other project I worked on was the Heap Game, where players are able to code, visualize, and play with heaps in a safe environment. My Contribution to the Heap Game was programming the challenges and the gameplay interface.

Background

There has been a drop in enrollment in Computing, and the Game2Learn program seeks to address this trend by motivating and retaining students. Our goal is to give students an alternative way of learning by introducing games as learning tools. Students report less intimidation in the class, and teachers notice improved enthusiasm among their students. Game2Learn games provide students with interactive visual representations of common computing concepts.

Figure 1: A screenshot of the first gameplay challenge from Heap Game.

Research

Our first task was to get everyone used to coding in C# using the XNA Framework, by creating a Pong game. Once we were all familiar with C#, we began getting the Heap Game ready for a study. We sat down with our advisor and talked about our goals, set deadlines for the game, as well as some design issues.

Heap Game uses the Darkwynter Engine, a game engine created by UNC-Charlotte students for the game studio class. The game engine features many helpful functions for creating a game such as displaying models, handling input, modifying the terrain, and an in-game compiler which can be used to write and run programs in game.

We created the first coding challenge for the first level of the Heap Game, as well as the gameplay challenge for the first level. The coding challenge allows players to populate the heap in the game world with values, shown by height in the terrain. The gameplay challenge allows players to reorder the heap into a proper max-heap by swapping values up the heap, using the button interface. We also created the coding challenge for the second level which calls the participant to fill in a method stub for inserting values into a heap.

With one challenge complete, we created an in-game pre and post test to evaluate possible learning gains among participants. Though the game was not complete we could use the pre test and the first level to evaluate the game's ease of use and the test's difficulty.

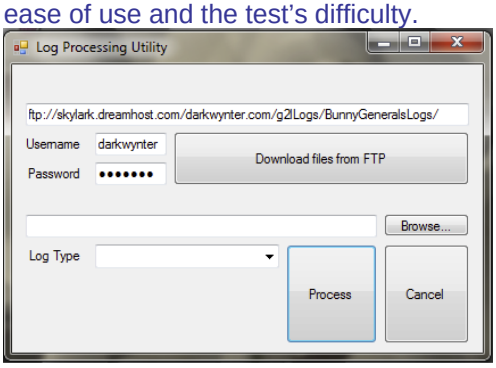

Figure 2: The Log Parser used for Heap Game and Bunny Generals

We also created a logging system which would create log files for player actions within the game. Also, it creates log files from the pre and post tests and sends the files via FTP to our server to be parsed. The log parser we wrote, also in C#, downloads the files from the FTP server and parses all the game logs into one file, with each entry separated by player number. It takes the test log files and places them into an Excel spreadsheet to be graded, and their statistical information extracted.

Impact

- **Heap Game's use of an in-game compiler allows it to be an extraordinary learning tool, allowing players to visualize correct and incorrect code.**
- **From preliminary observation of people interacting with the Heap Game, it seems clear the game has potential to effectively teach about heaps.**
- **Those working on Heap Game have learned a great deal about working in three dimensional environments and about learning in general.**

Conclusions

Since all the levels are not completed we cannot evaluate learning gains as of yet. However, allowing people to playtest and observe the challenges involved in the first level we were presented with excellent feedback for improving the game.

Those working on the game gained greater experience with C#, learned about how to use a game engine effectively, and about working together in a team efficiently. It can be difficult working with someone else's code, but once we became familiar with the workings of the Darkwynter Engine we were able to get things up and running rapidly.

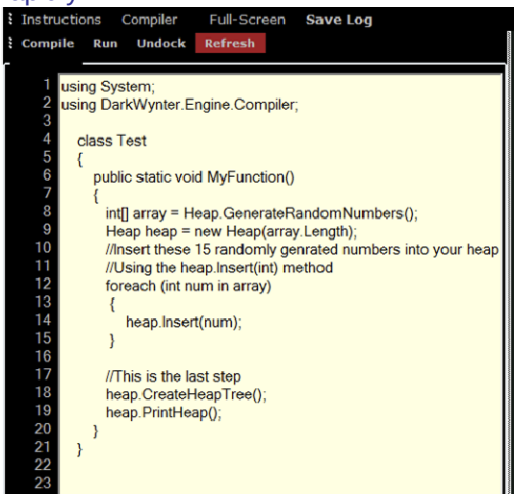

Figure 3: A screenshot of the second level's coding challenge from Heap Game.

Future Work

Currently, Heap Game contains one complete level, and a coding challenge for the second level. We plan to have three complete levels, each with a coding challenge and a gameplay challenge. We also plan to have people playtest along the way so that the finished product will be much more polished than if we only test amongst ourselves.

The biggest challenge will be to create gameplay challenges that effectively inform the player about the construction of heaps and maintaining the heap upon removals. Finally, we will run a user study with the finished product and evaluate the results for learning gains, as well as user feedback.