CSC111 Project 2 Proposal: Python Alchemy
A Graph-based Crafting System

Yianni Culmone, Omid Hemmati, Neyl Nasr, Benjamin Gavriely

March 5, 2024

Problem Description and Research Question

Many video games have crafting systems. Vastly popular games like Minecraft, Terraria, No Man's Sky, Subnautica, and Stardew Valley contain different mechanics, motivations, and outcomes. Regardless of all their differences in implementation, games with crafting systems appeal to players interested in creativity. (Grow et al.)

For games with crafting capabilities, the crafting system is often a keystone of depth and cognitive challenge. Crafting systems allow players to create new items/elements, advance in the game, and pursue creative endeavors. The appeal of crafting lies in its satisfaction of discovering new recipes, solving puzzles of resource harvesting and management, and tackling problems through unique methods (such as different items/powers/recipes). Either way, an appropriate inclusion of a crafting system in a video game would result in an enhanced depth and replayability of the game. (Grow et al.)

Crafting, due to its inherent roots in creativity, can be used as an effective learning method. It is a multi-sensory experience that can cater to various learning styles. Visual learners benefit from seeing and understanding how different components fit together, and tactile learners benefit from solidifying abstract concepts into concrete examples. There is a positive correlation between creativity and academic abilities, particularly in areas like reading, comprehension, and writing tasks. Fostering creativity within an education system can improve academic performance (Tzachrista et al.), so promoting creativity within a video game's crafting system can be seen as beneficial from most standpoints.

The goal of this project is **to implement a crafting system using graphs to foster creativity, and turn learning into a tactile gaming experience.** This problem matters because understanding the underlying mechanics of crafting systems can allow for the creation of a more engaging, creative, and educational gaming experience. By focusing on a simple model using graphs (associations between vertices), we can optimize and enhance the components that make crafting appealing to a player's creative side, such as a lack of limitations/rules on what a player can craft. This model can then be used in other games, itemsets, and applications, resulting in a multitude of use cases.

Computational Plan

Graph representation:

Our project, essentially, is a game engine used for creating a crafting model. By definition, an engine should be able to manipulate different datasets. Due to this property, we will be building our game with the intention of loading a set of crafting 'recipes' from a csv file. (More on how we will source this file later) Using a file reader, a graph will be created to represent the recipes. Each item (ingredient or final result) is represented as a vertex on the graph. The name of the element will be the item attribute of the vertex. The neighbours of the vertex will be represented as a dictionary. This is a modification of the Graph ADT, to accordingly accommodate the needs of our engine. In the dictionary, two groups are represented. Group 1 encompasses the required ingredients to craft the item. Group 2 represents the recipes in which the item is involved. Through this, the two cases are represented: What it needs to be crafted, and what can be crafted with it. This model is quite similar to how books and reviews were represented in Exercise 3, where the elements linking the vertices of the graph together are not as obvious as just reading from the file and will require further iterations and computations.

We intend to auto-run a function to create all the vertices and edges of the graph as the csv file is being read, in accordance to the above rules.

Real world data set:

To display our engine's use cases, we want to represent its usage in a form factor that anyone can understand and appreciate. In order to meet this condition, we decided to take inspiration from the crafting recipes of the game Little Alchemy (Claiborn et al.) As we now know, alchemy is a very limited representation of how the elements in our world interact. However, that makes it very easy to understand so that the true potential of our game engine can be perceived without complexities. The dataset contains 570 crafting recipes, (Claiborn et al.) however we will alter it to allow for a more up-to-date representation of our 'default recipes', for a total of approximately 600 crafting recipes. Obviously, these recipes are just a default use case for our engine and can be altered to infinite lengths through modifying the provided csv file.

Sample Data: (obtained from IGN.com, cited in References)

```
Acid Rain, (rain, smoke)
Acid Rain, (rain, smog)
Air, ()
Airplane, (bird, steel)
Airplane, (bird, metal)
Alarm Clock, (clock, sound)
Alcohol, (fruit, time)
Alcohol, (juice, time)
Algae, (water, plant)
Algae, (sea, plant)
Algae, (ocean, plant)
```

Key Observations:

From this sample of our dataset we can see some key points about our data:

1. Some items have multiple crafting recipes
2. Some items do not have any recipe (granted to the user by default)
3. Manipulating this data to fit our ADT will require many computations.

Additionally, this data is not formatted in this form in the dataset. The dataset is originally represented as a table, which is less useful for a Python implementation. This format will likely change, but this is a simple representation of how it might look.

Game Interface:

To display our game interface, we plan on using pygame for graphics, including cosmetic modifications like a background, start screen and music.

To customize our game's appearance with backgrounds and start screens, we can utilize Pygame's `pygame.image.load()` function to import assets. (Pygame Intro)

For music, we will use the `pygame.mixer.music.load()` and `pygame.mixer.Sound()` functions to play an engaging soundtrack throughout the game experience. (Pygame Intro)

We will also use text boxes/ dropdown boxes to allow user input of elements. Once the user enters a valid combination of elements, a text field will appear that keeps track of all new elements generated, a notification to the user of the new element discovered, and adding this new element as an option in the text box.

In addition, we will be displaying the statistics of the player's score, such as longest path and time, for example, using visual representations of the data. This allows the player to keep track of their 'progress' while playing the game, as a form of automatically generated feedback.

Finally, through our main menu, we intend to display a plethora of modes.

Modes:

Mode 1: Edit mode:

Although the game's recipes can be edited in the csv file, some players may prefer a GUI to use. Due to this, we will provide a mode that allows the user to edit the recipes of the game, and automatically save/push the changes to the csv file through the usage of a file writer.

Mode 2: Flashcard mode:

In order to further seal the identity of our engine as an educational tool, we will include a flashcard mode. Just like its name suggests, it will allow the user to guess the result of a randomly selected craft, which will be very useful for learners who enjoy using word association.

Mode 3: Chemistry mode:

We intend to include a version of our engine which uses real-world chemical reactions in place of recipes. Additionally, this mode will allow the user to simulate a reaction with more than two "ingredients", unlike the normal mode. We intend to represent a large number of commonly used chemical reactions, which will also be compatible with the flashcard and edit modes.

References

Claiborn, Samuel, et al. \Little Alchemy Cheats - List of All Combinations
- Little Alchemy Guide - IGN." IGN, 17 Feb. 2021,
www.ign.com/wikis/little-alchemy/Little_Alchemy_Cheats_-_List_of_All_Combinations.

Grow, April, et al. \Crafting in Games." Digital Humanities Quarterly, 2017,
www.digitalhumanities.org/dhq/vol/11/4/000339/000339.html.

Pygame Intro | Pygame v2.6.0 Documentation. www.pygame.org/docs/tut/PygameIntro.html.

Tzachrista, Maria, et al. \Neurocognitive Profile of Creativity in Improving
Academic Performance|A Scoping Review." Education Sciences, vol. 13, no. 11, Nov. 2023, p. 1127.
https://doi.org/10.3390/educsci13111127.