15-112 Term Project Design Analysis

*Tentacle War*

Shaojie Bai

**I. Brief Introduction**

This game is characterized by its strategic nature and clever AI (designed to act similar to a user--- try to occupy other cells). The user tries to assimilate enemy cells, whose AI also tries to defeat the user. The most important details of my code design is divided into these sections:

1. Important functions of pygame: pygame.mixer.Sound() for music, screen.blit() for background images, and pygame.sprite for collision. These are important features to determine things like collisions of EMB cells. Moreover, there are efficient built-in functions such as pygame.mouse.get\_pressed(), which returns which of the three buttons on the mouse get pressed. Also, pygame runs much faster than Tkinter, which makes my program--- which needs to run lots of things per round, able to be played smoothly.

2. Important functions of pickle: Module pickle is used to save and load data. Basically, it saves the data (only sequences) you want to record by outputting a .txt file. And next time when you want to load the progress you have last time, just import the .txt file and set a variable like self.achievement to it. By this way, pickle enables me to freely access previous records.

**II.** **Competitive Analysis User Interface Design**

One of the games on the Internet close to the **Tentacle War** that I am making is the "Cloud War"--- though it is a bit easier. I also import several images from the Internet, which will be in the Work Cited document. The game "Cloud War" can be found at the website below:

<http://www.kanogames.com/play/game/cloud-wars>

And here is one of the screenshots I take of the game:

Figure 1. Cloud War

There are a few very good features that will be quite useful to my game design:

(1) When you drag a mouse from one cloud to the other, a blue line "travels" with the arrow such that it is easier for users to see what is going to be the target.

(2) As a cloud becomes stronger, its size gets larger. I am considering something similar, but using color instead.

(3) It has a good Artificial Intelligence (AI) in that the enemy cloud can attack the neutral / (my) friendly cloud with clever decisions.

However, there are also bad features that I try to avoid.

(1) Bad interface. I made my interface more beautiful than this, and the help menu and achievement page clearer. This game is not an easy one to play with, so help menu is really important.

(2) Tutorial design. Instead of simply putting a "help," I include both a help menu and set my level 1 as the tutorial level. At this level, there are captions guiding users to perform certain actions. Such design can help users quickly master the basic skills required to play this game.

(3) Life Value. As is shown above, the cloud contains no life value, so it's hard to see when it is assimilated and when it is in danger. By including factors such as "maximum life value" and displaying each cell's value on it, the strategies the users take to play the game will not be disturbed and thus, better.

(4) Tools. There are needles in my game that the use can use to help assimilate enemy cells. Tools are usually an important part of a game, but I didn't notice it in competitive analysis.

***--- Interface Control***

In general, I use a modified version of MVC (Model-View-Controller) in this program. But instead of Tkinter, I use pygame. The MVC model controls the interface displayed during the game: background, cell, the animation of those decorating-purpose curly lines around a cell, and the needle. Also, based on my conclusion from competitive analysis, my main menu interface, help menu and level-choosing menu are displayed in a clearer way with clear guidance. These are constructed using images and relevant functions in my program, instead of MVC. Considering this game is new to most of the people, I think a clear interface is necessary.

***--- Feedback From User Experience Lecture (Tuesday)***

In the user experience lecture, a few people told me that without carefully follow the help menu--- which normally people don't read carefully--- it is hard to play this game. And I was suggested to add a tutorial level. This is a very constructive suggestion as it can be quite helpful to get new players familiar with the game. This new feature is added by making Level 1 the tutorial level, without any AI.

Also, one other feedback is that the distribution of AI difficulty should be adjusted. Thus, I make the first 3 levels relatively easier by letting the AI be simple.

**III. Other key functions design in my own program**

2. Identify Chains. I have a function tryMoveATT(self), which uses the mouse click information to determine which chain should be created. The chain is a class. At first the chain.shouldGrow is set to be True; and after the chain reaches the target, it is set to be False, and the signal transferring begins.

3. Transfer in chain function: traceTransfer(self) . This is another important function in my program. Basically, for every existing chain, it performs the following steps. First, by checking the life value of the source cell of a chain, program automatically adjust the frequency of transferring life--- strong cells transfer more quickly. Then, by looking at the state of a chain, such as chain.shouldCollapse, chain.shouldBreak,etc., the function determines the current state of a chain and thus the subsequent moves. Also, when two chains of the same color are created between two cells (e.g., a green chain from A to B and one from B to A), the first connected one collapses back.

3. Artificial Intelligence. I use the principle of "Finite State Machine" in Computer Science. Basically, this is a method that, by checking each individual of a sequence, determines what state it is in--- like "attack." In each call of self.timerFired(), enemy cells are prompted to make decisions. By searching through their surrounding environment (enemy cell class have function think(self)), like what's the state and strength of their allies and enemies, enemy cells establish priority queues and pop out targets to deal with.