**Think Aloud 1**

This week, I decided I want to start creating a simple game in C++. I’m still deciding what kind of game I want to make, but I think it will be something along the lines of a tic-tac-toe type of game and see if I want to expand to something larger. From there, I’m thinking of possibly trying to work on a “tanks” game where players take turns shooting at each other’s tanks over a barrier. I’ll be starting with the following tutorial: <https://www.simplilearn.com/tutorials/cpp-tutorial/game-in-cpp>. I’m excited to work on this project and see ideas come to life.

**Think Aloud 2**

This week, I’m getting started on my C++ game. I decided to start off with tic tac toe. I found a couple of tutorials that could help me get a grasp on the idea of how things work with developing a small C++ game. I’ll paste the ones I’m going to use a start for what I want to base mine off of. So far, I’ve created my repository and my tic\_tac\_toe.cpp file. I’ve also started writing pseudocode, and I plan to start following the tutorial with the coding next time. I’m excited to see how this goes, and I hope to gain more of an understanding of the backbone behind C++ based game engines and games.

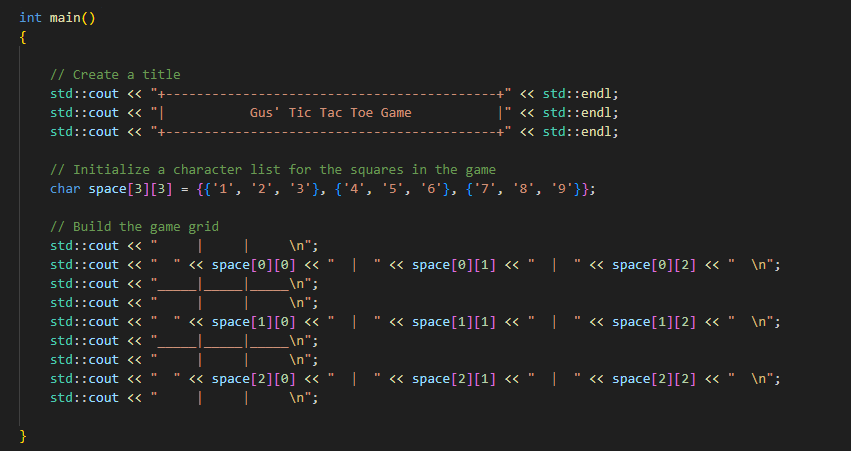
Tutorial Links:

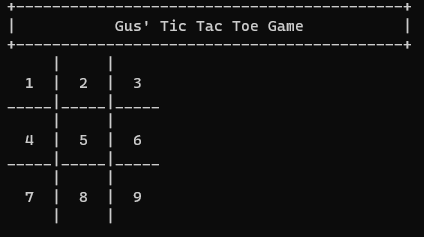
<https://www.simplilearn.com/tutorials/cpp-tutorial/game-in-cpp>

<https://youtu.be/OuEHYoCHGUQ?si=2AL3i2MLGJ2AeeGS>

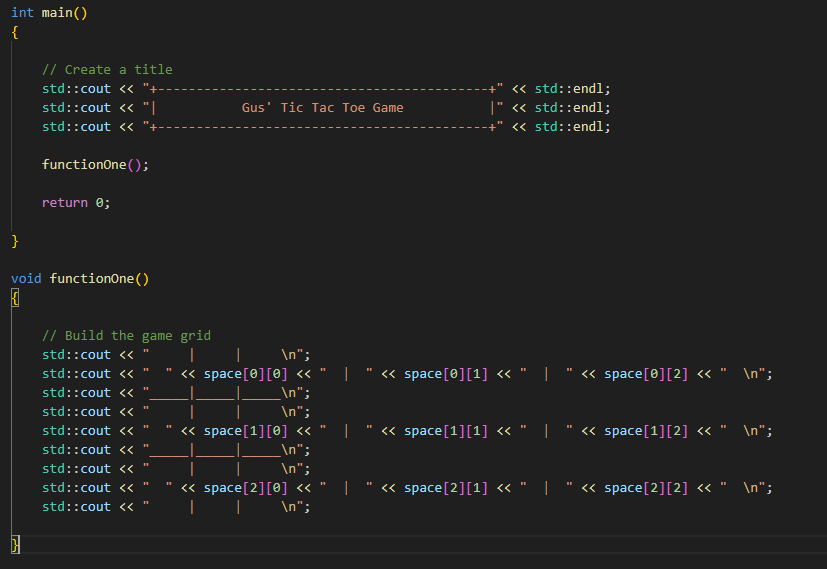
**Think Aloud 3**

This week, I started adding code to my pseudocode. I started by adding a table that uses a character set to fill each square in a formatted tic tac toe game with a number from 1 to 9. I’m still following the same tutorial for the game for now and will add my own input and create a new game from there.





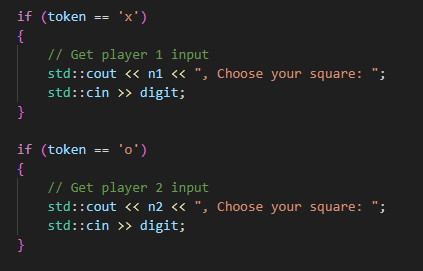
Next, I separated the title and grid into different functions. The program’s output still looks the same as it did, but now it’s more organized and the right step toward OOP. I’m now working on understading the logic of the decision making for the user to choose which box to choose and how the number is replaced with an X or O.



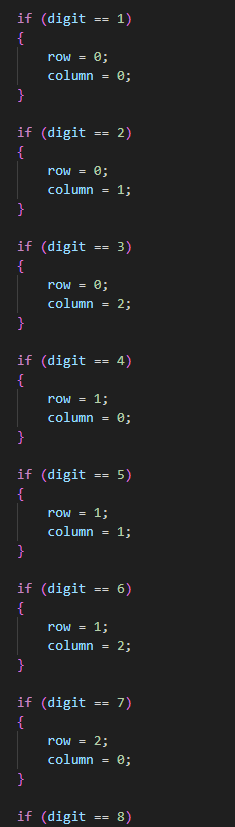
I’m excited with the progress I’ve made, even though it’s not a whole lot yet. I’m mostly trying to understand the logic right now before I start diving into the actual interaction part of the game. I hope to eventually make a game with more of a user interface and possibly objects that can be moved by the user. It would be cool to better understand how to use C++ for games with interfaces, not just CLI programs.

**Think Aloud 4**

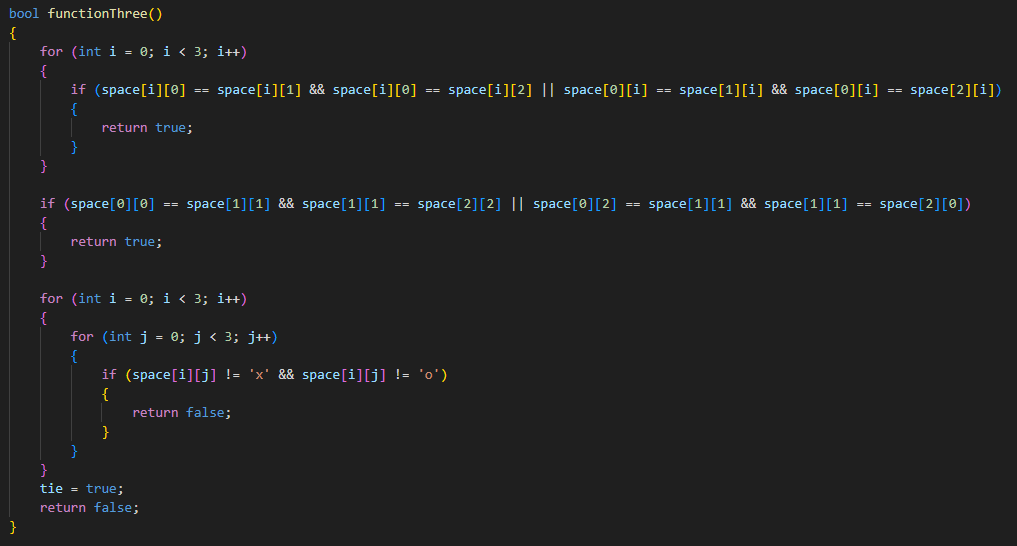
This week, I added the game logic from the tutorial. I am starting to understand how the game works at a fundamental level and some basics in simple game design. I started by adding input prompts for each player in my functionTwo(). This used if statements so it would ask corresponding players for their input when it was their turn.



This was a simple concept enough concept to grasp as it was just if statements and input. Then, I added if statements based on choices so the row and column would be set to a value based on the user choice. This determined the placement of the x’s and o’s.



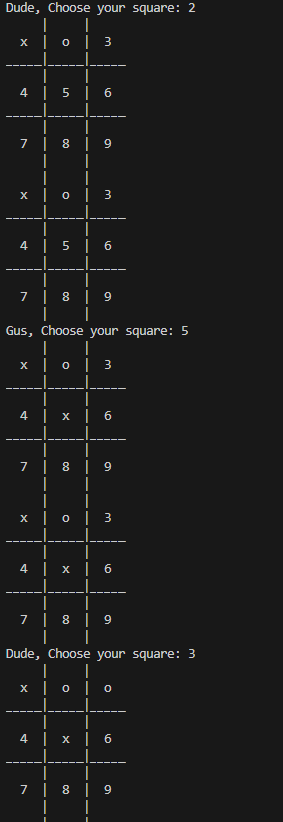
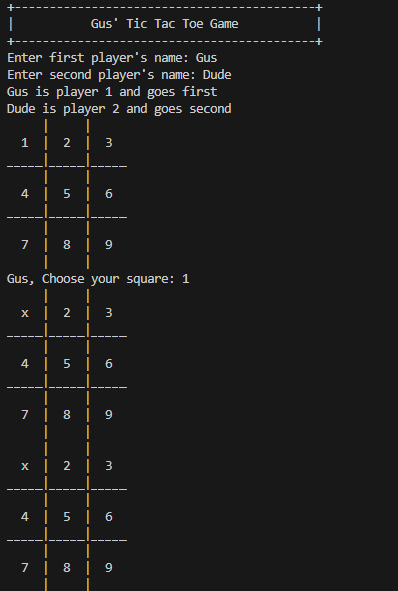
After this, I added some input validation for values outside the range from 0 to 9. Then, I added if statements for placing the x’s and o’s in the chosen squares. This was done by testing whether an x or o is already in the square. If one isn’t, it replaces what’s in the square with an x or o depending on which player was entering data. I then added a functionThree() that acts as a game loop by testing to see if the game has ended or not. It checks for horizontal, vertical, and diagonal wins as well as draws.

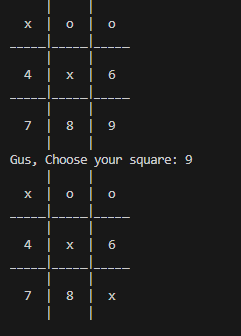


Finally, I added player names input by users and a loop of the functions to run the game in the main function.



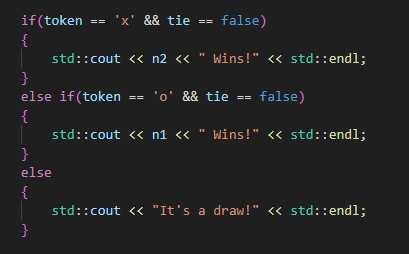
Right now, the game functions well even though I haven’t added a winner or loser message part of the game. The only thing I have to figure out in terms of fixing right now is that the game grid prints twice whenever a player enters a choice. I’m sure this won’t be a very complex fix, so I should have it figured out soon. I’m enjoying working on this project, and I’m excited to add my own ideas into the game and make it my own as well as expand to other games when I tune this one up. I’m thinking of finding out how to replace the current grid when a player enters instead of printing a new one. I’ve done this in python, so I’m curious to see how to do this in C++.



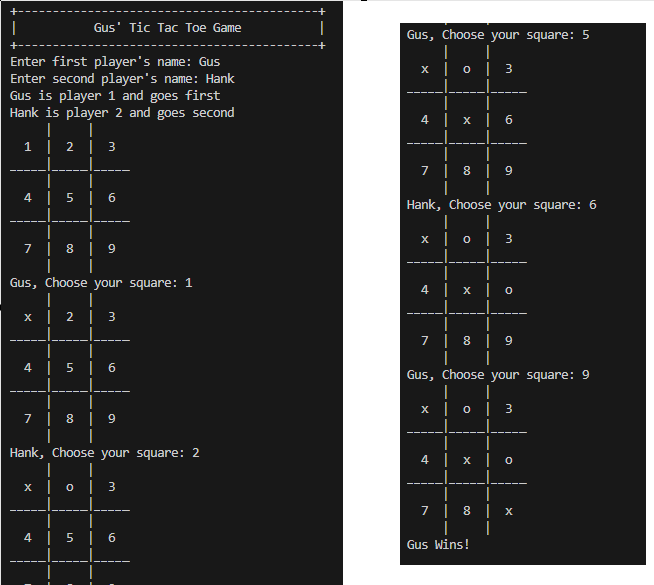


**Think Aloud 5**

This week, I started by separating some parts of the program into methods. I did this with the title portion and getting names for the players. These are called in the main method. I then went on to add the win statements that end the game. This was simple as it was just if else statements for the different cases.



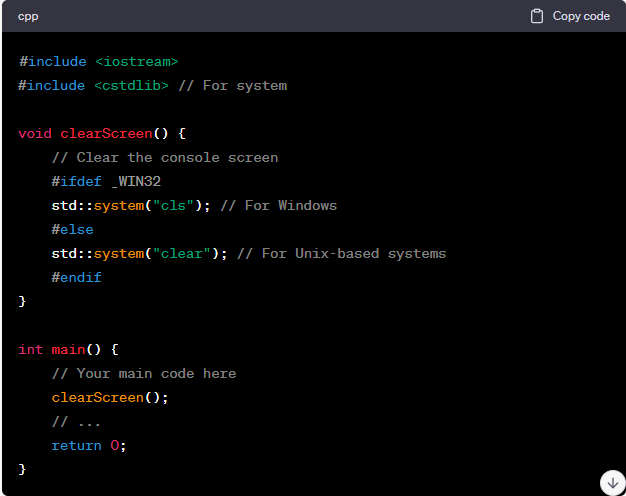
I then changed the arrangement of function calls a bit as I noticed that functionTwo() calls functionOne(), so I took functionOne() out of the loop in main(). I moved it above the loop, so the game grid would only display once each time through the loop and once before the loop.



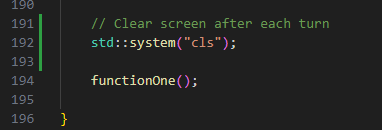
I didn’t change a whole lot this week, but I was happy to see some fixes made that caused my game to look and function the way I wanted it to. I completed this tutorial, and I’m now going to move on to learning how to replace the current game grid with the new one instead of printing a new grid beneath every time. This will give me a better understanding of drawing of game surfaces works in C++.

**Think Aloud 6**

This week, I decided to follow a Chat GPT tutorial on how to get text to replace previous lines so the game can look more like a game and less like a menu. I asked how to clear the screen in C++, and I was given a simple response.



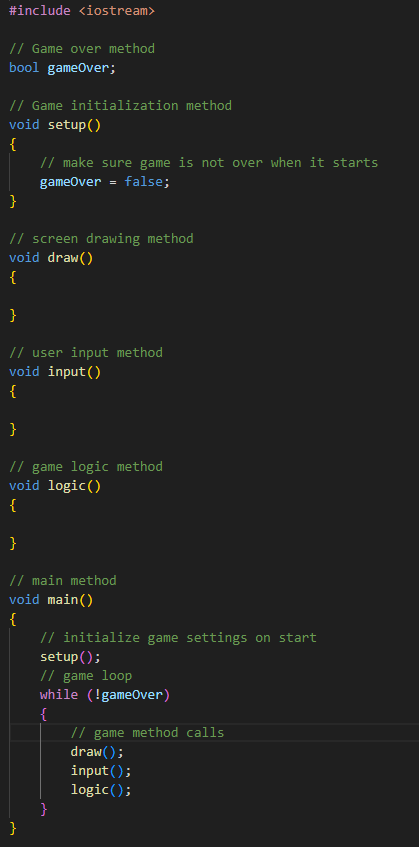
Since I know I’m only running this on Windows, I decided to just add “std::system(“cls”)” to my functionTwo() so the screen clears right after a player takes their turn.



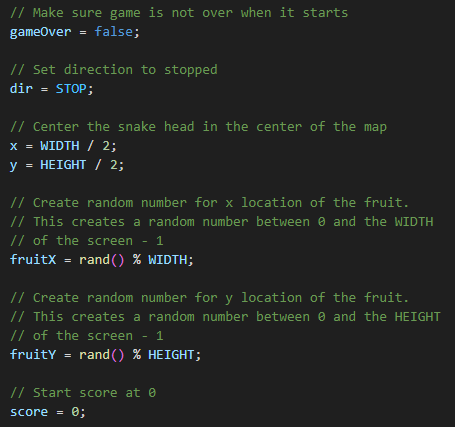
This was a pretty simple and easy fix, and I noticed that the game isn’t responding right to a tie and is instead continuing to ask the players to make their choice when the board is filled. I think this will be the next problem I address with the program before possibly moving on to my own game idea.

**Think Aloud 7**

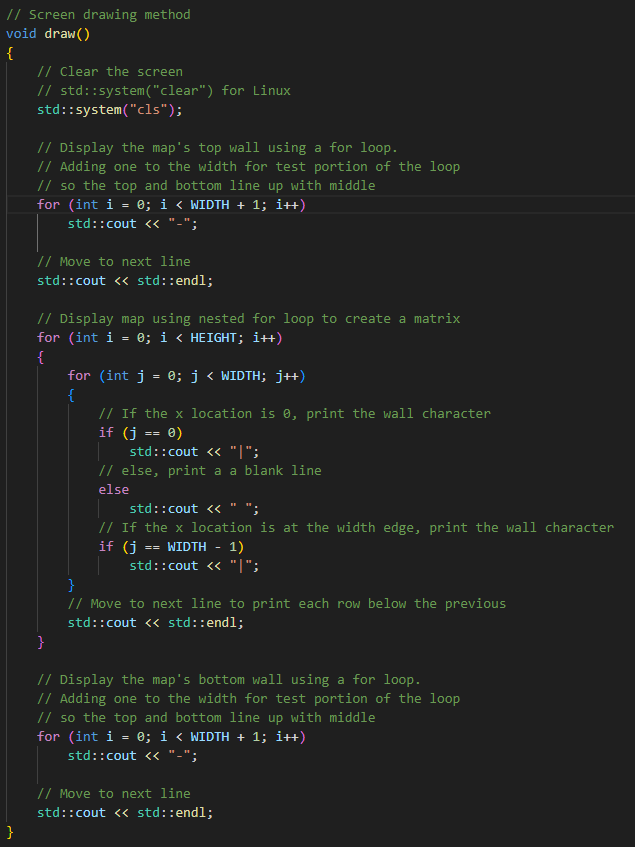
I’ve decided this week that I want to start making more of a graphical game. I’m going to start developing an extremely simple snake game. For this game, I will be following the following YouTube tutorial: <https://www.youtube.com/watch?v=E_-lMZDi7Uw>. This is the first of a three-part playlist to completing the game. I started by adding my methods that I plan on using for the project. I followed the tutorial on how to create a basic game skeleton.

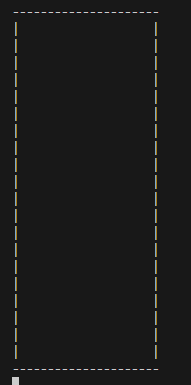


I then added variables for different aspects of the game.



Next, I added code for drawing the game to the screen. Right now, the game is not optimized well at all, and the map is currently really slow and laggy when being drawn to the screen. It does work, however, which is my main focus right now in following the tutorial and building an understanding of how visual games work.

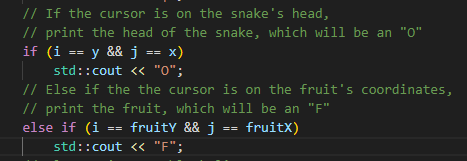




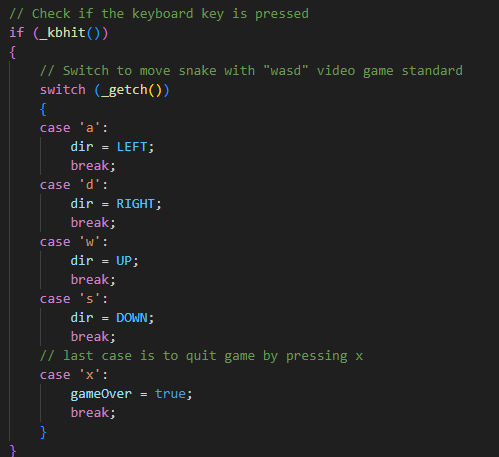
I’m enjoying this tutorial more, and I understand how it works a bit better than the previous one. I feel confident in moving on to the next tutorial soon, and I’m excited to see how this snake game turns out.

**Think Aloud 8**

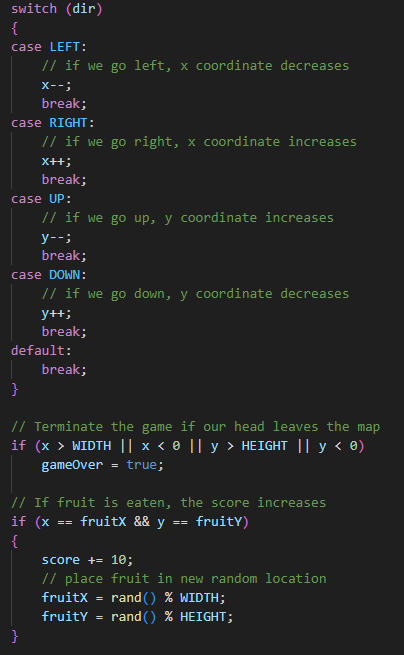
This week, I’m following the second of the three videos in the snake tutorial. I started with making new cases in the nested for loop to print the snake’s head and the fruit.



I then added a switch to set up the input with “wasd” for controlling the snake in the game.

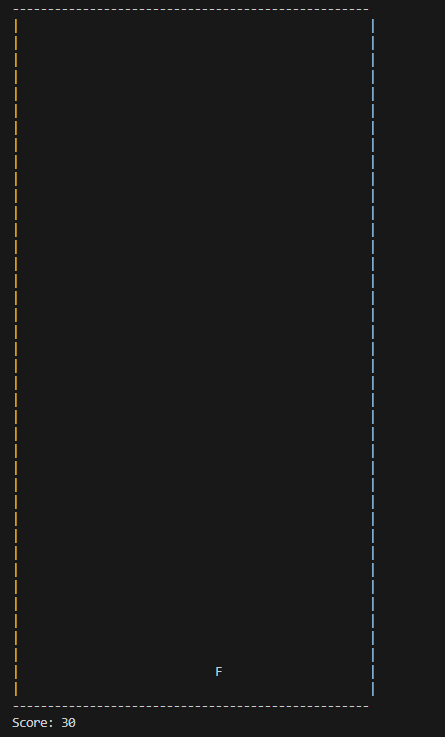


Then, I added game logic that determines where the snake moves with different input, and a portion that ends the game if the player leaves the map as well as a system for collecting fruit and adding to the score. I had to add a section to print the score in the draw method for this to display.





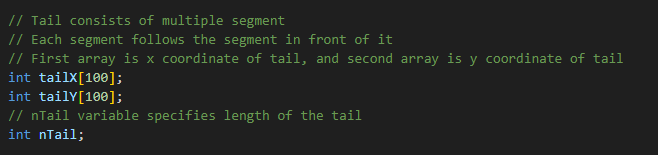
My game now shows a head that can be controlled with the “wasd” keys, and the user can collect fruit that spawns in a new location every time it’s collected. The score increases as well.



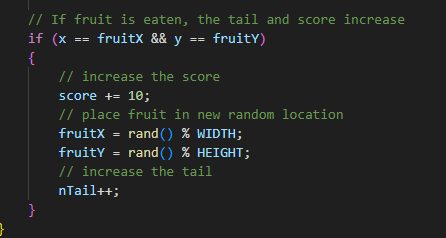
Overall, I’m happy that I’m learning game logic and input for movement in the game. I feel like this is helping my understanding a bit about how game development works in C++.

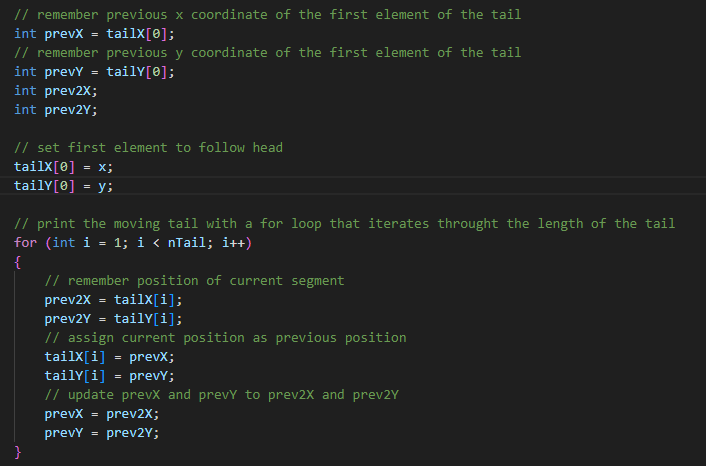
**Think Aloud 9**

For my 9th Think Aloud, I finished the tutorial on the snake game. I followed the third tutorial, which is located in the following link: <https://youtu.be/PSoLD9mVXTA?si=Ur484WmPnLL-2Mq4>. This tutorial is about adding the tail of the snake to the game. I started this by adding two array integers for the coordinates of the tail and an integer that specifies the length of the tail.

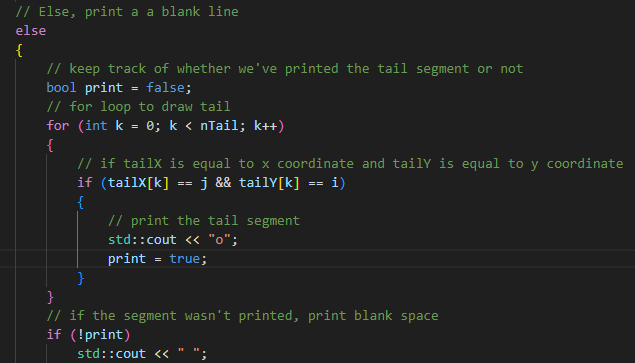


I then added the game logic to print the tail as fruit is collected with a for loop, and the tail increases as fruit is eaten.

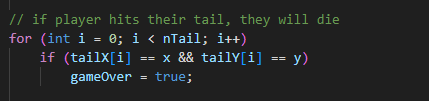


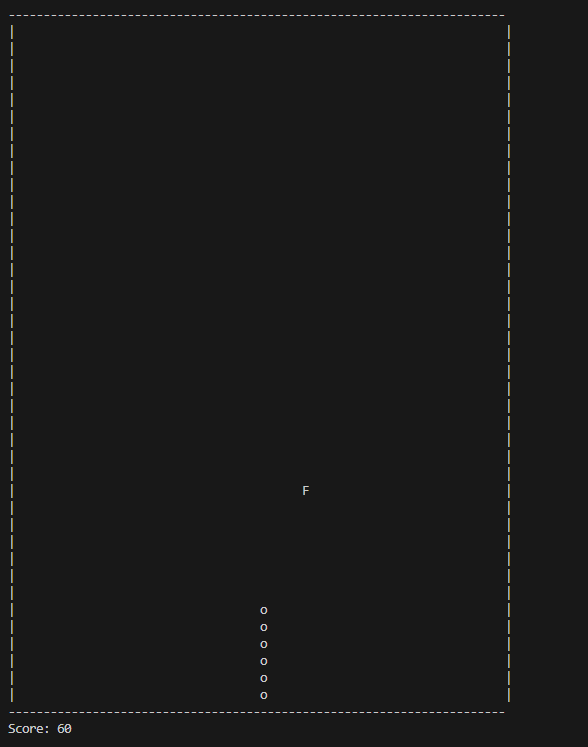


Next, I added the drawing portion of the game for the tail, which required another for loop to draw an “o” when needed and a space when needed. This needed a for loop to iterate through each element of the tail array.



Finally, the game is fully functional, and at the end, I added his suggestion for making the player lose if they hit their own tail.





So far, my high score is only 60 because I haven’t played much, but I’m happy to see the game functioning, and I now have a much greater understanding of some simple game principles in C++. I’m going to try adding some features to this and checking out some more concepts to keep expanding my knowledge of game development fundamentals. Let me know what scores you can get in this game.

**Think Aloud 10**

For my 10th Think Aloud, I decided to add some quality of life improvements to the snake game. I’m adding a game over screen that displays the final score. I first added a game over method that would be called when the game ends. I then added a system clear portion as well as the score in the middle of the screen.

A screen shot of a computer program

Description automatically generated

I then called the method after the game loop so the game over screen would display when I died.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

This week gave me a little more insight as to how the game works with its loop and how to display a new screen outside of it to make a game over screen. I’m happy with how far I’ve progressed in understanding simple C++ games, and I hope to expand on this and possibly make a C++ game with a GUI next semester.