Snake ‘97

(Snake’22)

Slither.io

By:

Riya Deokar (riyaa)

Camila Zhan Jin (czhanjin)

Ruhaan Bhagat (rbhagat)

Zakarey Sharif (zsharif)

EC327 Summer 2022

Github: https://github.com/czhanjin/EC327-Project

Video: https://youtu.be/B2iaftpaee8

Task Distribution and Assessment of Effort

Member and Role	**Tasks**

Ruhaan Bhagat
(Specification Lead)
Assessment of Effort: 25%
	- Check design specifications. Problems encountered: (1) screen sparkling, (2) board failed to report desired score, (3) image of the apple did not load.
	  - Fixed by:
	  (1) Window.display() was commented out. Fixed by taking out the comment.
	  (2) Initialize the board index.
	  (3) Adjusting the pixels and dimensions of the images.
	- Music command, stop button.
	- Report: title page, task distribution, software architecture, overview.
	- Video making.

Zakarey Sharif
(Interface Lead)
Assessment of Effort: 25%
	- Add comments throughout the code and make sure it is readable.
	- Make sure code is visually pleasant, crisp, intuitive and robust.
	- Game over command.
	- Check design specifications. Problems encountered (1) Message would not display on screen. (2) Program does not exit when borders are touched.
	- Fixed by:
	  (1) Create a separate window for the game over text.
	  (2) Use four different if conditions for each of the boarders of the board that when satisfied, exits the program.
	- Report: task distribution, overview.
	- Video editing.

Riya Deokar (Project Lead) **Assessment of Effort: 25%**	- Tracking and documenting timeline, milestones, objectives. - Create the option to have 1 snake (1 player) or 2 snakes(multiplayer).

	- Make sure code is efficient and secure. - Score board, movement direction, size of the snakes. - Check design specifications. Problems encountered (1) Unable to create 2 snakes. (2) Snakes did not move in the direction specified by user. (3) Score board recorded the wrong numbers. - Fixed by: (1) Create two separate classes for each snake and two different functions for each of the snakes. (2) Assign a number to the direction of snake movent in the functions. Call the number in the main function (instead of doing so in the specific functions). (3) Since the score counts the size of the snake, score = fruits eaten - snake initial size. - Report: task distribution, component description. - Video making.
Camila Zhan Jin (Technical Lead) **Assessment of Effort: 25%**	- Check progress, code testing. - Create and set up Github and ReadMe. - Main structure (classes, and randomize fruit). - Check design specifications. Problems encountered (1) Snake would not increase in size when it touches the fruit. (2) Fruit were displayed in fixed places. - Fixed by: (1) Creating a class for snake and fruit, and a function that relates the two. (2) Randomize the position of the class fruit. - Report: task distribution, component description, software srquitecture. - Video editing.

Timeline and Planned Agenda

Date, Time and Location	**Topic**	**Duration**	**Members Present**
Friday, June 24th, 2022 Vitual: Zoom	Come up with project ideas	1 hour	All members
Saturday, June 25th, 2022 Ingalls	Game chosen initially: pac-man (did not work out)	3 hours	All members
Monday, June 27th Classroom	Create snake, basic commands of the game (changing direction, moving, imputing board, snake, fruit images), debugging, report	6 hours	All members
Tuesday, June 28th Classroom	Make snake eat the fruit and become bigger, 2 players, add music, game over command, pause button, debugging, report.	8.5 hours	All members
Wednesday, June 29 Classroom	Finalize project: debugging, video presentation, report	5 hours	All members

Overview

Marketability

The classic Snake'97 game was developed in the late 90s. The goal of our project is to create a new version of the original game that is more user friendly. The original game has a dot-matrix, black-and-white display with monotone sounds added in the background. Our Snake'22 not only has all the functionalities of the 97 version but also has a colorful display, relaxing background music, a scoring board and also an option to let multiple users play simultaneously. Our goal is to allow users born in the 2000s and 90s and earlier to take a break from reality, go back in time and get a good dosage of nostalgia while also having a better playing experience.

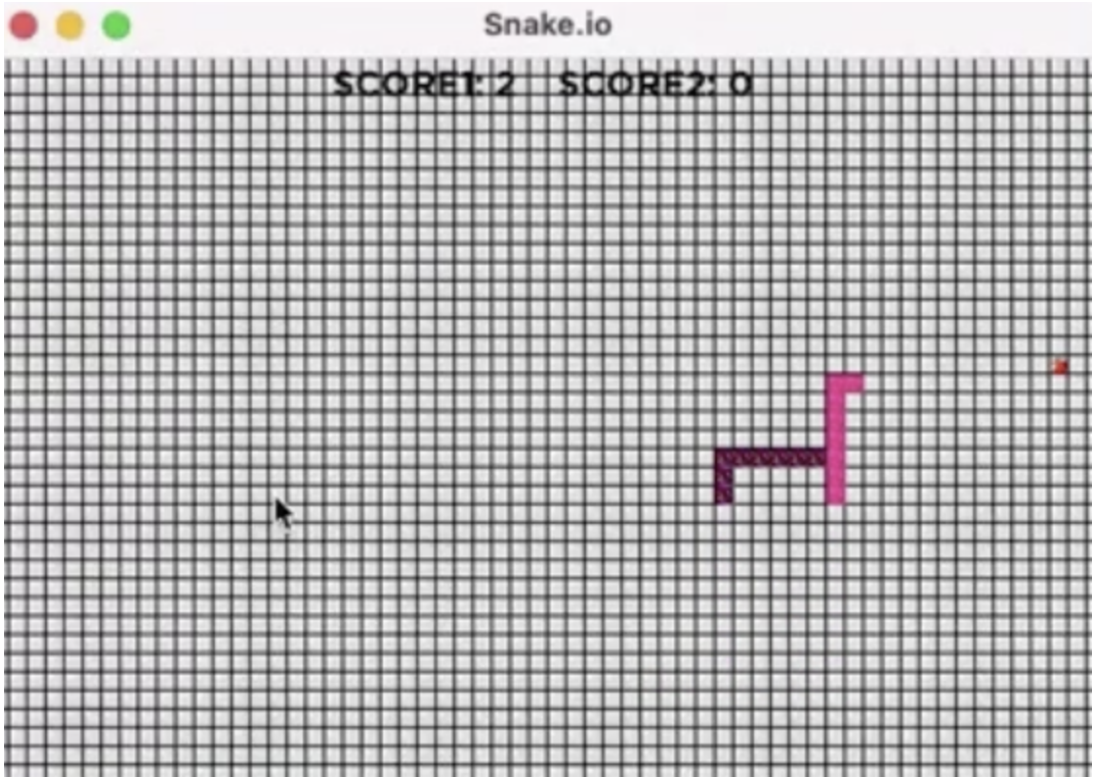

Fig 1. Improved Snake game with new features incorporated.

Software Arquitecture

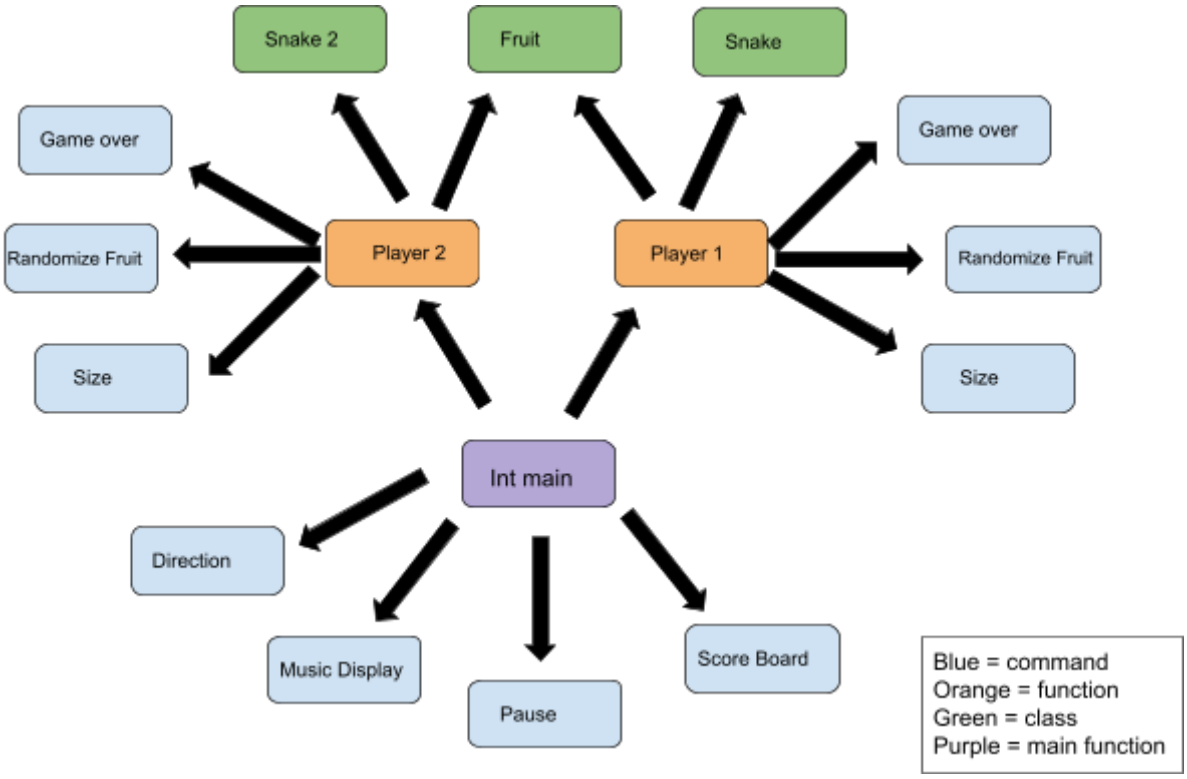

Fig 2. The program consists of 4 commands called in the main function and 2 functions called from int main. Each function performs 3 commands on 1 shared class and 1 different class.

Front End

- Our game is crisp and intuitive. Once the code is compiled and run, the user selects the number of players and can start the game.
- Our project does not crash and only exits the game when borders are touched.
- Visually and audibly pleasant. Unlike the original black and white snakes, we made our snakes sparkly and colorful. Unlike the monotone sounds in the original game, we added FCB Barça's anthem in the background to uplift the players.

Back End

- Our code runs efficiently and smoothly. It does not fall in an infinite loop and it exits as soon as the user loses the game. It is secure and can not do anything malicious.

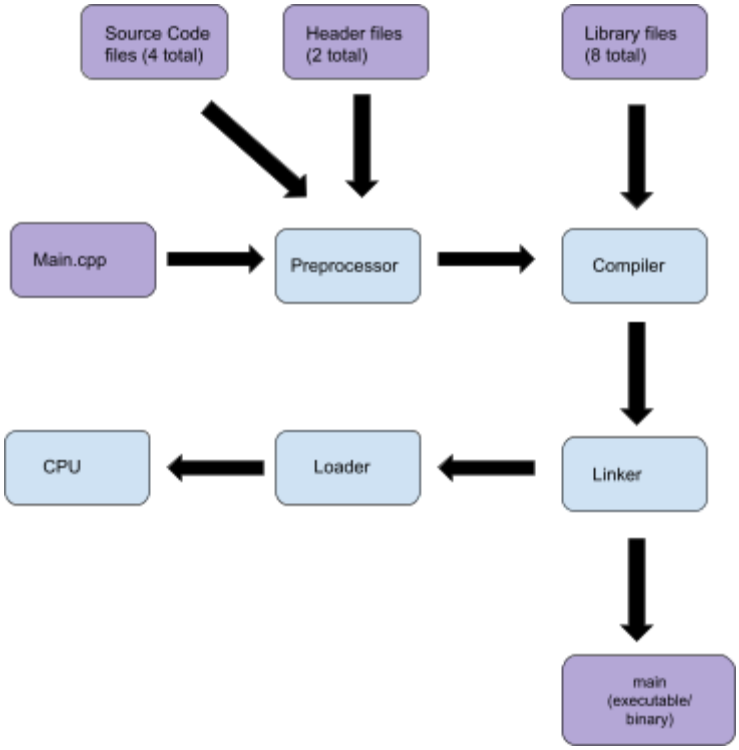

Fig 3. Basic structure of the program. Process by which Main.cpp is converted into a computer executable binary file main.

Component description

We first created a rectangular board with customized height and width size, and three classes (Snake, Snake2, and Fruit). To facilitate the calling of the classes, we created two functions (Player 1 and Player 2) that can individually perform the basic commands of the snake such as moving in any possible direction (up, down, right, left), increasing the size when the snake eats a fruit, randomized fruit position (class Fruit) on the board, as well as customized commands such as displaying game over messages and quitting the game 10 seconds after when the snake moves beyond the board. The purpose of creating two separate functions is to allow class Snake and class Snake2 to perform their job individually but also simultaneously when both functions are called in the main function. This allows the user to choose between single or multiple player game.

Inside the scope of the main function, we added a music file that is played from the beginning of the game until the program is exited. Additionally, based on the defined int variable number in the movement section of the functions Player1 and Player2, we have assigned a keyboard key to each of the directions of the snake movement (up, down, right, left and A, D W, S). The main function is structured in a way that classifies the commands based on single or multiplayer. We used if conditions based on the user input to call Player 1 or Player 1 and Player 2 simultaneously. We also have a command that allows the user to pause the game when key P is pressed by introducing a while loop that when conditions are satisfied, the program sleeps and prints out a message for the player. To unpause the game, the user only needs to press P again. Last but not least, we have created a new window that displays the score of the player/players in real time. The score displayed is the number of fruits eaten by the snake and it is done by counting the size of the snake minus its initial size.

Code: Compiling and Running

(1) To compile the code, input the following command in the terminal:

g++ main.cpp -o main -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio.

(2) To run the code, input the following binary file in the terminal from step 1:

main

Note: in order to successfully compile the game, user must:

(1) install sfml

(2) Import image directory, music file and conio header file and the main.cpp c++ file and place them inside the same directory

(all files can be found at: https://github.com/czhanjin/EC327-Project).