Data Structures and Algorithms

PROJECT REPORT

CODENAME PROJECT: WARGAMES KAI

Prepared by:

Stanlly 2301923533

Mika 2301923962

Gadtardi 2301929480

Even semester 2020

Computer Science Program

Binus University International

Team Name: The Goomers

Team Biography, Roles, Responsibilities, and Contributions:

We formed our team out of mutual respect and recognition of our passion and skill in harnessing the power of modern computing. Our goal in this project is to appreciate the potential of data structures and algorithms in solving real world problems. Our team's strengths lies in our passion, skill, and eagerness to learn about Data Structures and Algorithms in programming. Additionally, we believe our love of videogames and our belief in the potential of AI and other future technologies will also contribute greatly in our efforts to do this project.

Gadtardi Pratama Wongkaren:

Gadtardi believes in life-long learning of new skills and talents. He believes that computer science will be more crucial in the coming century or more and that we as young people need to rise up to the challenge and understand this new phenomenon of technology in order to create our brightest possible futures.

Role: Project Manager, Quality Assurance, Documentation

Responsibilities: Final Report, Project Documentation, Progress Reports, General Management

Contribution: Topic Research and Finding, Progress Reports #1, #2, Project Documentation, Code Analysis, Final Report and Presentation

Mika Mahaputra:

Mika is an avid gamer and particularly loves to appreciate impressive implementations of artificial intelligence in video games. In particular, he loves observing how AI in games can continuously defeat human opponents. It is his life-long mission to be able to defeat good AI in games and he does this by living the concept: "To defeat your enemy you must know him"

Role: Lead Programmer of Vector Version, Secondary Programmer, AI Enthusiast

Responsibilities: Creation of the Vector Version Skeleton and Making sure it works

Contribution: Wrote the code and debugged the Vector Version of the Snake Game, Documentation

Stanlly:

Stanlly sees gaming and programming as among two of his great hobbies. Stanlly's passion in games heavily involves studying the inner mechanics of how a game works, including its inner workings. In his leisure time, Stanlly often likes to program as he describes it as "calming".

Role: Lead Programmer of 1-D Array Version, Debugger

Responsibilities: Create the main program as well as the 1-D Array Version of the Snake Game

Contribution: Menus, Colors, 1-D Array Version of Snake Game, Debugging for both Versions, Documentation

Team Goal and Problem Description

Our Group Motivation

Our group is motivated by the idea of being able to work on a project that is unique to our group.

As stated in our project proposal, as avid lovers of gaming, we are motivated to find a project that is related to games or games development using data structures.

We hoped that we could create multiple different programs that can then be compared to each other after we had a better understanding of the code and structures involved.

On Mika's enthusiastic request we also hoped to also implement AI into our program but that concept has since been scrapped

The Problem Statement:

What kinds of data structures would be the most efficient for the creation of a Snake Game in C++?

Strategy to Tackle the Problem:

We would look online in order to gain a better understanding of how to implement a simple version of the Snake Game.

After we have had a better understanding, we would try to create as many versions of it as possible in order to compare the efficiency and implementations of each version along the way.

Additionally, potentially creating an AI for the project in order to add more unique features to our project.

Possible hindrances in the way:

Before jumping into this project, we expected a few issues to hinder our progress:

1. Due to the pandemic, we were not able to fully work efficiently as a group. Our schedules tended to conflict and caused us to have some time constraints

2. We were expecting some of the different data structures used to create the Snake Game to be harder to implement than what we are used to

Investigation and Data Structures Chosen

Investigation over types of Game:

We scoured the internet for ideas. We had a few suggestions from our seniors' previous projects: Anywhere from mazes to minesweeper and even tetris.

However, we wanted to find a topic that could prove a challenge to us.

Eventually, we ended up going with the idea of creating a snake game. We started off with wanting to make 2 versions as well as potentially adding an AI to play our game.

However, due to time constraints, we were not able to implement all of the features that we were planning.

The End-Result Software Problem Statement:

After a lot of discussion, we ended with this current statement:

What kinds of data structures would be the most efficient for the creation of a Snake Game in C++?

We had a few hypotheses that we wished to try out:

- Vectors
- Linked Lists
- 1D/2D Arrays

Another scratched idea was:

Can we create an AI that can play the Snake Game till the game ends

But due to time constraints, this never made the final cut.

Why we chose to do a Snake Game:

We first considered a maze game as our first idea. Over a few sessions of discussion, we eventually decided to work on a snake game.

We thought that the idea of making a functional snake game using data structures would be a fun and unique challenge for our project. Last years' batch had groups doing maze games already.

We hope that our choice of doing a snake game would be unique for our batch and provide an interesting challenge and concept.

Why we chose 1D Array:

1D Array was chosen due to its simplicity.

- Each coordinate of the snake can be stored in an array

- The efficiency of the code would in theory also be decently fast
- 1D Array doesn't require the creation of a structure or a new data type
- These arrays can then be used to easily check whether the snake tail has to be drawn

Pros:

1. Easy to debug
2. Simple to understand and implement
3. No need to create a custom data structure
4. Base Code is very robust and isn't bug prone
5. The longest processes are for checking snake coordinates

Cons:

1. The simplicity makes it hard to implement complicated features
2. The need to iterate through all indexes of the arrays for certain processes may not be very efficient

Why we chose Vector:

Vector was chosen because of its (theoretical) better adjustability.

We hoped that vectors would be a far better data structure for more robust code as well as better overall processing speeds and potential to be more adaptable than 1D Arrays.

The implementation of the Vector version would require:

- Implementing a new structure in order to store our data
- This structure would then be used to govern the different segments of the snake
- Each segment will contain the coordinates of the snake and can be deleted when needed

Pros:

1. Faster performance
2. Can store more array and objects
3. Much more simple memory management
4. Potentially infinite snake and game length

Cons:

1. A lot harder to implement
2. The code is very prone to bugs and glitches
3. A few bugs that is hard or couldn't be fixed

Planning and Design

Physical End Product Objective:

Our plans for the end product was to have at least 2 fully functional versions of the Snake Game.

This would let us present 2 different data structures as well as how they are implemented and the differences between each process.

Pros and Cons of each method would then become clearer as we finish implementing it and analyze each different method.

Proposed Structure for end application

Menu:

- Menu that lets you select which game to play
- Create a menu that is fully functional and aesthetically pleasing
  - Has colors
  - Formatted in a readable way
- About Page
- Control Scheme Information Page (No Settings)

Desired Results for end product:

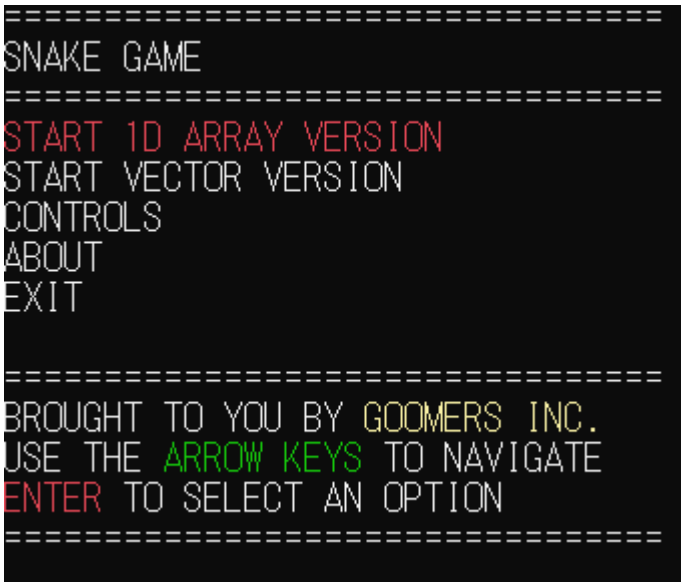
- Create a fully functional standalone snake game
  - Let the snake move through a wall and appear on the other side
  - The snake increases in size every time it consumes food
  - Snake is killed upon colliding with itself
  - Score is kept at the bottom of the screen
  - The boundaries should be clear
- Be consistent with the theme and be in color
  - Aesthetically pleasing
  - Good Formatting
- Let it be executed from the Main Menu
- Exit the game to the Main Menu once the game has finished

Program Manual and Code Execution Results

Link to Code and Video:

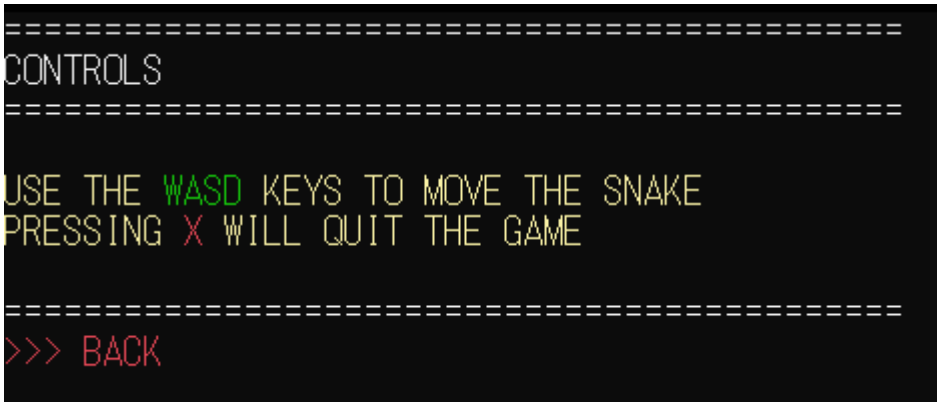
https://github.com/CitrinePiasora/GOOOMERS
https://www.youtube.com/watch?v=cPzYxkqWYxc

Main Menu:



The Arrow Keys are used to navigate through this menu. The appropriate selection shows up as red when it is selected. Enter a specific menu by pressing Enter

Controls:



This screen explains the controls for both versions of the game. WASD are used to change the directions of the snake

About Page:

```
============================================================
MEET THE CREW
============================================================

STANLLY: LEAD PROGRAMMER AND DEBUG TESTER/FIXER
 PROGRAMMER OF 1D ARRAY VER

GADTARDI: ORGANIZER, DOCUMENTATION AND QUALITY ASSURANCE

MIKA: SECONDARY PROGRAMMER WHO STRIVES TO WIN AGAINST AI
PROGRAMMER OF VECTOR VER

============================================================
>>> BACK
```

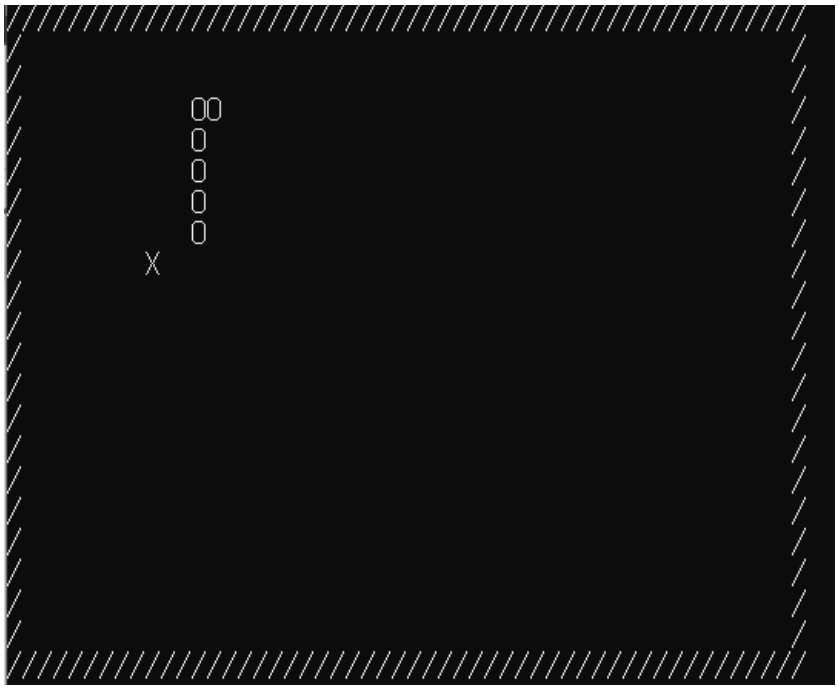
The team behind this game

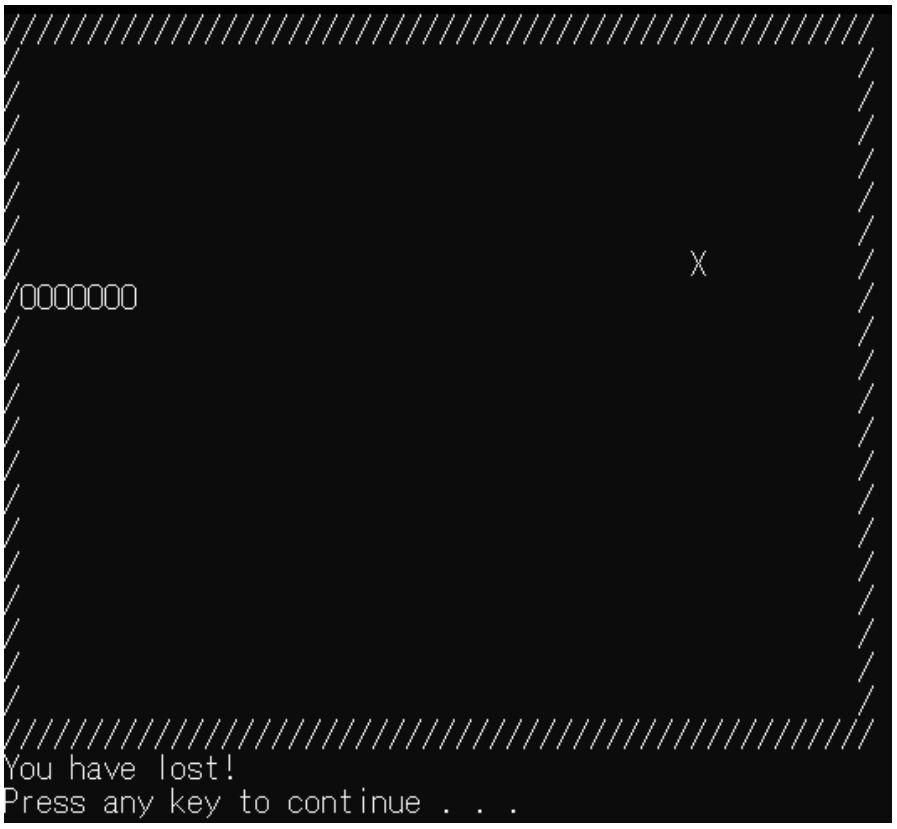
Snake 1D Version:

```
########################################
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#        o                             #
#        o                             #
#        o                             #
#        o                             #
#        o                             #
#        o                             #
#        o                             #
#        ooO                           #
#                                      #
#                                      #
#                                      #
#                      F               #
#                                      #
#                                      #
#                                      #
########################################
Score:90
```

```
########################################
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
#                                      #
# oooooooooO                           #
#                                      #
#                                      #
#                                      #
#                      F               #
#                                      #
#                                      #
#                                      #
########################################
Score:90
Press Enter to Return to Main Menu
```

Snake Vector Version:


OO
O
O
O
O
X


X
OOOOOOO
You have lost!
Press any key to continue . . .

Program End Result and Evaluation

Flaws in 1D-Array version

Flaw 1

It's not very flexible. There's only a finite length that the snake can get to and it uses 100 Array memory in order to achieve it

Flaw 2

The program has to process every value in the array from the head to the end of the tail, this may be a lot slower as the snake starts to increase in size

Flaw 3

It has a tendency to break around score 330. Food stops spawning and there has been no fix that can be found

Flaws in Vector Version

Flaw 1

The game has a problem with updating its coordinates (X, Y) preventing the player from passing through the walls.

Flaw 2

The code is rather hard to implement when compared to other data structures such as 1D-Array and Linked List.

Flaw 3

Finally the game won't reset the snake length when the player loses which will require the player to rerun the code or exe files in order to reset the length.

Regarding the program

As a whole, we feel that we have reached very close to our end goal. We have made 2 fully functional versions of the Snake Game Idea that are ready to be presented.

Due to the lack of AI, we aren't quite fully done with our original statement, but we feel that we are close enough to solving the problem and understanding which data structures are efficient and easy to use when making the Classic Snake Game.

Possible Improvements to program:

There are a few things that we can improve on:

- Fixing the numerous bugs found in the Vector Version

  - Including but not limited to:

    a. Inability to pass through walls.
    b. Adding color to the Vector Version.
    c. Coding Issue: Y coordinate of snake breaks, preventing the ability to pass through walls.
    d. Inability to create a loop that resets every time the game is started (Currently requires a reset of the entire program).

- Fixing the score limit of 330 in the 1D Array Version.

Possible alternative data structures used:

- Linked List

Regarding the Project Overall

The work process was mostly simple.

Stanlly and Mika would focus on writing and perfecting the code, while Gadtardi worked on documentation and helped give some Ideas as well as helped make sure that the game was working correctly and that there were no issues between both versions.

By the end of the project, we feel that we have all gotten a better understanding on how different data structures can have an effect on the efficiency and the complexity of the resulting application. Ranging from how it works to why it is used.

Team Reflection

Gadtardi:

Conducting this project has been a very rigorous challenge for me overall. There were many difficult and unpleasant challenges in doing this project that I found to be very challenging to me personally. However, I think that the process of doing this project has given me much valuable input, skills and insight that I think will be very helpful in molding me as a computer science student and being a better person in general, especially for any work I have to do in Binus computer science and elsewhere in the future.

Challenges I faced during this project:

- There were a lot of important issues outside of Binus schoolwork that required my energy and attention therefore a lot of my focus in studying the subject was diverted
- The coronavirus crisis forced a lot of radical adaptations in both my study process at Binus and life in general

Things I learned during this project:

- I learned how to work in a team more effectively, especially for much work required and tight deadlines. There were a lot of processes behind the scenes where I learned much when it came to collaboration.
- I learned to be tougher in general both as a student and as a person, as this project along with my other projects and situations outside of Binus work exerted a great deal of pressure on me physically, academically, mentally and perhaps spiritually too. I learned much about self-care and perseverance during this project I think will greatly assist me in the future as a student.

Mika:

My personal experience working on this project has been quite fun but riddled with many challenges.

The things that i liked when doing this project:
- We get to work as a team.
- As a person who likes playing video games it is fun to make and test our own games.
- We get to test and practice using multiple data structures to solve different problems.

However there are some difficulties that me and my team faced during this project:
- It's hard to communicate online (bad internet connection and etc) and its hard to meet up due to the pandemic.
- It's hard to implement certain data structures into our project due to lack of resources on the internet, and different data structures require some certain knowledge.
- Our goal to deliver a more polished and advanced version of our project is hard to reach since we have less time to study, to test, and to discuss the codes.

Stanlly:

Overall, there was a few big challenges along the way:
- Lack of time made it hard to meet up (especially due to the pandemic).
- The end result wasn’t as polished as I had initially hoped we could do.
- Time constraints made implementing every idea we had nearly impossible.
- There weren't many resources that we could look for in order to increase our understanding on how certain parts of the project could have been handled.

Despite all this, I walked out of this project with… Alright results. There were a couple of things that could've been better but overall, there's not a lot of things that went horribly wrong.