# HCMUS - University of Natural Science
# Faculty of Information Technology
# Project: Caro Game

**SID:** 20125110
**Name:** Le Pham Nhat Quynh
**Class:** 20CTT2

## 1 Introduction

This report includes all the techniques, algorithms used in this project, function description and their purposes as well as the compatible reference source.

## 2 Theme

All the music is inspired from game Undertale of Toby Fox due to the resemblance in the 8-bit appearance.

## 3 Game features

| Function | | | Function | Self-grading Point |
|---|---|---|---|---|
| Start Game | Play in P-P mode | | Start playing game<br>A variety of options:<br>+ P-P mode: human vs human<br>+ PC mode: human vs computer (3 levels) | 9/10 |
| | Play in P-C mode | Easy | | |
| | | Medium | | |
| | | Hard | | |
| Setting | Choose player icon (X/O) | | Custom the game according to player preference. | 9/10 |
| | Music (On/Off) | | | |
| | Size of the game board (3x3/5x5/7x7) | | | |
| Statistic | | | View player data<br>Delete player data | 8/10 |
| About | | | Provide information about the game programmer | 10/10 |

|  |  |  |  |
|---|---|---|---|
| Exit |  | Exit game and close the console window | 10/10 |

## 4. Project structure

Language: C++
Integrated development environment: Visual Studio 2019
There are two play modes:

- P-P mode: human vs human
- P-C mode: human vs computer
  + Easy
  + Medium
  + Hard

Control:

- Move: arrow keys
- Choose: Enter

Rule:

- End if any players win or there no empty square on the board.
- Player wins the game when:
  + 3x3: 3 consecutive symbols in any of 4 dimensions.
  + 5x5: 5 consecutive symbols in any of 4 dimensions.
  + 7x7: 5 consecutive symbols in any of 4 dimensions.

### 4.1 Algorithm

**P-P mode:**

- Players interact with each other through the keyboard.
- Function: control3x3() / control 5x5() / control 7x7()

**P-C mode:**

- Easy level (3x3/5x5/7x7):
  + Random base
  + Description: Random the move by the rand() function.
  + Function: randomPC()
- Medium level (3x3/5x5/7x7):
  + Heuristic algorithm (Medium score board)
  + Description: For each square, respectively check all 8 directions to calculate whether to attack or defend. Use attack-defend score arrays to load the attack and defend scores. For each direction, if there are n enemies, plus attack[n] score and do the reverse to the defend score. Afterall, choose the square with max points to give a decision.
  + Medium score arrays:

Defend_Score2[7] = { 0, 1, 9, 81, 729, 6561,59049 }
Attack_Score2[7] = { 0, 1, 2, 3, 4, 5, 6 }

Defend score array much more outweighs the attack score array. Therefore, even though this algorithm is more advanced than the random base, it is not optimal due to the lack of attack.

+ Function: Medium3x3/Medium5x5/Medium7x7

(Source: https://www.youtube.com/watch?v=k9xGToiEbG4&list=PLYv35bVS8O27SgtphnOvWXEkLfzlQmuV6&index=36&fbclid=IwAR3rN8nfUPLW-dh59iN3L61-Qoh4W3JDGLAnV3i4KHKpRhXRFyG5_uHJKfg)

➔ Hard level:

+ 3x3: Minimax – Alpha beta pruning algorithm.

Description: A **minimax algorithm** is a recursive algorithm for choosing the next move in an n-player game, usually a two-player game. A value is associated with each position or state of the game. This value is computed by means of a position evaluation function and it indicates how good it would be for a player to reach that position. The player then makes the move that maximizes the minimum value of the position resulting from the opponent's possible following moves. If it is A's turn to move, A gives a value to each of their legal moves. Alpha beta is the advanced algorithm of minimax with the purpose of prune all the unnecessary branches in order to reduce the loop as much as possible.

Function: Minimax3x3()

(Source:https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/)

+ 5x5/7x7: Heuristic algorithm (Hard score board)

Description: For each square, respectively check all 8 directions to calculate whether to attack or defend. Use attack-defend score arrays to load the attack and defend scores. For each direction, if there are n enemies, plus attack[n] score and do the reverse to the defend score. Afterall, choose the square with max points to give a decision.

+ Hard score arrays:

Defend_Score1[7] = {0,1,9,81,729,6561,59049 }
Attack_Score1[7] = {0,3,24, 192, 1536, 12288, 98304}

The logical score arrays based on Dr.Le Ba Khanh Trinh's research paper are able to tackle most of the difficult strategy in the game.

+ Function: Hard5x5()/Hard7x7()

➔ Note: About the issue why two algorithms are applied instead of one, at first, minimax-alphabeta algorithm worked well with 3x3 scale. However, infinite loop happened when the game was upscaled to 5x5 and 7x7. Therefore, Heuristic algorithm was used to tackle this problem due to its efficient running time.

**Result checking:**

- Check by columns
- Check by rows
- Check by first-dimension diagonals

- Check by second-dimension diagonals

### 4.2 Support function and technique:

- Move, hide and unhide the pointer.
    - + Function: Control(pointer) / HidePointer() / UnhidePointer()

  (Source: Nguyen Trung Thanh - https://www.youtube.com/watch?v=C7yokRqdd4o&t=1s)
- Background music, animation.
    - + Function:

      Music : Tap(), Enter(),…
      Animation: (using support function)
        Wherex() / wherey() : return the coordinator of the pointer
        GotoXY() : move the coordinator
        Textcolor(): change the text color
        Sleep() : temporarily stop the program in an amount of time.
        Clrscr() : Clear the screen

➔ (Note: this function is reported to be better than system("cls") in several forums and Nguyen Trung Thanh's recommendation)

- Set the console window (used in display result)
    - + Function:

      setConsoleWindow() : change the size of the console window

      fixConsoleWindow() : fix the size of the console window
- Read, write file technique (used in saved the player data)
    - + Function:

      saveInfor(): write data to file

      Statistic(): read file and display data

## 5 Data structure

### 5.1 Library

- iostream : display the information / get the input
- ctime : assist in random function (avoid the similarity in random function)
- conio.h : provide console-screen-interacting function (e.g: clrscr(), getch(),…)
- cstdlib : provide random function
- Windows.h : provide console-screen-display function
- mmsystem.h : provide music function
- iomanip : assist result-display function (e.g: setw(),setfill(),… )
- string: assist in some string-relating function
- fstream: read and write file (used in save player data)

## 5.2 Variables

### 5.2.1 Global variable:

- Define:
  - Size of console**:** consoleWidth = 142, consoleHeight = 35
  - Control the pointer: UP = 72, DOWN = 80, LEFT = 75, RIGHT = 77, ENTER = 13, ESC = 27
  - Color:

| | |
|---|---|
| Black | = 0 |
| DarkBlue | = 1 |
| DarkGreen | = 2 |
| DarkCyan | = 3 |
| DarkRed | = 4 |
| DarkPink | = 5 |
| DarkYellow | = 6 |
| DarkWhite | = 7 |
| Grey | = 8 |
| Blue | = 9 |
| Green | = 10 |
| Cyan | = 11 |
| Red | = 12 |
| Pink | = 13 |
| Yellow | = 14 |
| White | = 15 |
| Default | = 7 |

- Struct object :This type of variable make programming clean and smooth (consist of coordinator, icon, name, …)
- Enumerate variable turn = {play1,play2}: assist players in taking turn play the game
- Key: to evaluate the input from keyboard;
- Others : common state of program
- Array Arr[][]: save players' moves on the board to evaluate the best move.
- Defend_score1[7], attack_score1[7]: score array used for evaluating heuristic algorithm in hard level
- Defend_score2[7], attack_score2[7]: score array used for evaluating heuristic algorithm for medium level

## 5.3 Extra files

- Music files: implement music effect for the game
- Statistic.txt (Text file): save the player data