Assignment 1

Team number: Cyberpunk-hacking-minigame Team 13
Team members

Name	Student Nr.	Email
Yu Chen	2668777	yuna.chen@student.vu.nl
Berry Chen	2672919	z6.chen@student.vu.nl
Xiaojun Ling	2664286	x.ling@student.vu.nl
Yudong Fan	2662762	y.fan@student.vu.nl

Introduction

Author(s): Yu Chen, Berry Chen, Xiaojun Ling

Cyberpunk Hacking Minigame - Infinity is an extended version of the original Cyberpunk Hacking Minigame(which is a minigame derived from the role-playing game Cyberpunk 2077). We call the system we are going to implement the "Infinity" mode of the original game because only one puzzle needs to be complete in the original game at a time whereas, in our version, the player shall always be provided with a new puzzle when the previous one is finished as long as the time limit does not count down to zero.

In the original game, the main character V may get rewarded with different levels of eurodollars, materials, or reduced invasion costs according to the number of Daemons completed(Daemon: named from the original game which refers to a sequence of codes to be put in the buffer successively). In the absence of role-playing elements in the original game, these rewards become unmotivated for a user to continue to play the game. Therefore, to increase motivation, we modified the reward scheme and the final goal of playing the game. In our design, the player can be rewarded with 15 to 50 point scores and 5 to 15 seconds to be added to the remaining time for each completion of a Daemon according to a selected difficulty level. Once the remaining time becomes zero, the game finishes. Therefore the final goal of the game is to complete as many as Daemons you can to achieve a higher score.

The main type of user of the system shall be:

- *Gamer*: A user who plays the game and makes use of the features provided by the system.

The following terminologies and rules from the original game shall be respected(more details of the original rules can be found here):

- ***Code Matrix***: a 5X5 or 6X6 grid matrix that contains 25 or 36 code cells(e.g. "1C", "E9") and performs as a selection pool for the gamer to pick codes from.

- ***Buffer***: A container that has a limited size. Code picked from Code Matrix shall be added to the Buffer in order.

- ***Daemon***: A list of codes waiting to be "uploaded" successively into the Buffer.

- **Puzzle**: A complete set of the puzzle contains one CodeMatrix, one Buffer and several Daemons.
-

- ***Rules for completing Daemons***: There are one to four Daemons in one puzzle. To complete a Daemon, the user shall pick cells one by one from the Code Matrix and should try to choose the code with the same string value compare to the codes in Daemon. The Buffer displays the codes picked in the picking order. A Daemon is complete and marked as “SUCCEEDED” if every code in this Daemon matches the codes in the Buffer in the same order without interruption. A Daemon is marked as “FAILED” if the remaining empty places in the Buffer is less than the remaining codes in the Daemon which are waiting to be added into the Buffer.

- ***Rules of selecting code***: At the beginning of solving a puzzle, only the first row of codes in the Code Matrix is selectable. The gamer shall only be allowed to pick one code from this row. After a code is chosen by the gamer, this code becomes unselectable in the matrix which means it shall not be selectable again during the game. The string value of the code goes into the Buffer in the order form left to right. Also, the selectable area changes from row to column and column to row after every selection. If the player chooses a code from a selectable row, then the selectable area switches to the column of codes that share the same column with the picked code, and vice versa.

The different entities from the CyberpunkHackingMinigame shall be as follows:

- ***Buffer Size***: A customized offset shall be added to the default buffer size to allow the user to adjust the difficulty of the game. The game is easier with a larger buffer size because the gamer has more chances to pick codes to match Daemon with Buffer.

- ***Timer***: Unlike the original game, the remaining time is dynamic and shall be bound to the reward system. A SUCCEEDED Daemon brings more time limit and a FAILED Daemon decrease the time limit.

- ***Reward***: Each success of completing a Daemon shall be rewarded with more remaining time and scores based on the difficulty the user chose.

The main modules of the system shall be as follows:

- ***UI***: The game window where the main user I/O operations take place. The graphical user interface(GUI) shall be used to visualize key entities of the game, including Code Matrix, Buffer, Daemons, Timer, Scores and a Menu which can start the game with different difficulty level.

- ***Game***: This module shall be the main module that deals with the user’s inputs, check codes in the buffer, reset remaining time and refresh new puzzles.

- ***Entities***: This module shall be the storage that stores puzzles and the highest score.

The game shall work as follows:

1. The user runs the game and selects a difficulty at the Menu and start a new game. If no difficulty is chosen, the game shall be loaded with a default difficulty level.
2. A puzzle shall be displayed in the game window and once the user picks the first code in the Code Matrix, the Timer starts to count down, and the Buffer and Daemons works follow basic rules of the game.
3. The game ends if the time counts down to zero or the player press the END button. The player can then choose to close the game window or back to the Menu to start a new game with other difficulties.

Features

Author(s): Yu Chen, Xiaojun Ling, Berry Chen, Yudong Fan

Functional features

ID	Short name	Description	Champion
F1	Commands	In the menu, the user may click the corresponding button to activate the following functions: - *set difficulty* - *start game* During the game: - the user may *select codes* from the code matrix by clicking the left mouse button - click “END” button to finish game - click “MENU” button to back to menu The player shall be able to close the game window at any time by clicking the “cross” on the top right of the application window.	Y.Chen
F1B	Undo	**BONUS** During the game, the user may click the “UNDO” button to undo one latest move. The player can use UNDO once every 10 seconds. This is to prevent malicious scoring by keeping the process of undoing the last step or steps to complete a certain Daemon to obtain the rewards of SUCCEEDED multiple times. We do not think the game should have the feature REDO because, in our design, UNDO is not a function the player can use without restriction. Given that the player has only one chance to UNDO in every ten seconds if REDO is allowed, the “precious” UNDO chance is then wasted. The scenario to use REDO is therefore very rare.	/
F2	Puzzle	A randomly chosen puzzle containing a buffer, a code matrix with matching daemons shall be displayed on the game window when the game starts. A new puzzle shall be loaded if the remaining time is not zero and every daemon has been rewarded. The source of the puzzles shall be stored in .txt files.	Fan
F3	Difficulty	There shall be four difficulty levels for the user to choose from. Different levels shall be different in the buffer size, initial time limit, time rewarded and score rewarded.	/
F4	Basic play rules	Code selecting rules and Daemon completion rules shall respect the original game listed in the introduction section.	Ling/Berry
F5	Rewards	10 seconds and scores from 15 to 50 shall be rewarded for	Y.Chen

		every successfully completed Daemon corresponding to different levels of difficulty. There shall be a penalty with a deduction of 5 seconds for each Daemon that fails to complete.	
F6	Timer	There shall be a timer that starts to count down after the user picks the first code from the Code Matrix. The game shall be over if the remaining time becomes zero.	/
F7	Scores	The current score and highest history score are displayed on the game panel. The most recent highest score shall be encrypted and written into a save file and displayed on the game panel. This feature is used to assess the performance of the player and provide motivations for the player to continue to play the game.	Fan

Quality requirements

Author(s): Yudong Fan

ID	**Short name**	**Quality attribute**	**Description**
QR1	Pre-game validation	Reliability	Text files that contain valid puzzle data shall be parsed by the **Parse** and saved in the **Puzzle.** The parsed puzzle data must adhere to its corresponding text file. Any exception in this process shall be caught, and an error message shall be delivered to users.
QR2	Speed of dynamic interactions and changes	Responsiveness	All the dynamic interactions and changes need to be done within 100ms after the user chooses a tile. The dynamic interactions and changes include: - Disable this specific tile from choosing again - Add the chosen tile into the buffer and display it - Switch the opening row or column
QR3	Speed of loading new puzzles	Responsiveness	During the runtime of the game, loading and delivering a new puzzle need to be done within 200ms after the current puzzle is exhausted(currently there is time left **and** there is no available daemon).
QR4	Definite Code Matrix Cell	Reliability	Each cell in the code matrix is identical and unique by its coordinates regardless of their values. When a cell in the code matrix is selected by the user, it is exactly the selected cell that is removed from the code matrix and added to the buffer. This ensures the logic operations are based on the actual cell rather than some other temporary or copied values.
QR5	Rapid and functioning menu	Usability	All the clickable buttons that appear on the menu page can be accessed by users anytime when the menu is opened. Clicking on a clickable button shall lead to its corresponding behaviour (e.g. set difficulty or start the

			game) within 200ms.
QR6	Encrypted save file	Security	The save file is encrypted with base64 encoding to prevent easy editing of the highest score from users.
QR7	Undo and Menu buttons	Availability	During the game, the use of the **undo** and the **menu** button needs to be under certain restrictions according to the game rule. Only when the requirements for using the buttons are met, the two buttons shall stay activated and become clickable. Otherwise, they shall always be shut down from clicking.
QR8	Traceable game frame	Maintainability	Each frame during the game is snapshotted and saved. This makes every interaction between users and the system traceable. Therefore, changing in status can be seen clearly. Adding, altering, deleting logic operations shall be easier to do.

Java libraries

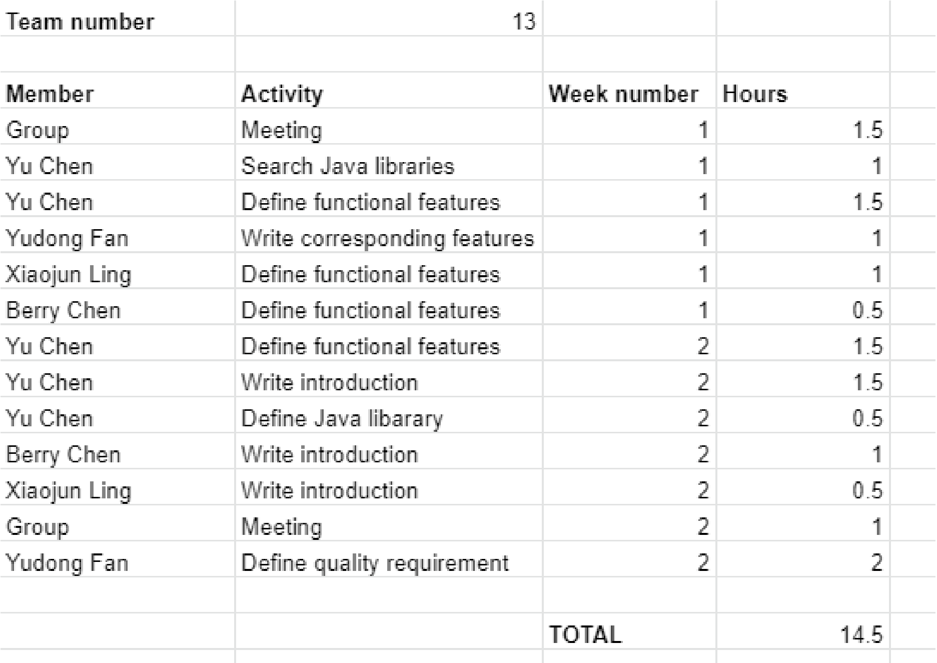
Author(s): Yu Chen

Swing

Used for styling the user interface of the system. We chose it among others because Swing is from the JAVA official library, which means no extra efforts for importing the libraries are needed and the learning materials are rich. It is an old library but can fulfil all of our needs.

Time logs

<Copy-paste here a screenshot of your time logs - a template for the table is available on Canvas>



Team number	13		
Member	**Activity**	**Week number**	**Hours**
Group	Meeting	1	1.5
Yu Chen	Search Java libraries	1	1
Yu Chen	Define functional features	1	1.5
Yudong Fan	Write corresponding features	1	1
Xiaojun Ling	Define functional features	1	1
Berry Chen	Define functional features	1	0.5
Yu Chen	Define functional features	2	1.5
Yu Chen	Write introduction	2	1.5
Yu Chen	Define Java libarary	2	0.5
Berry Chen	Write introduction	2	1
Xiaojun Ling	Write introduction	2	0.5
Group	Meeting	2	1
Yudong Fan	Define quality requirement	2	2
		TOTAL	14.5