Assignment 1 - Hacking Minigame

Team number: 14
Team members:

Name	Student Nr.	Email
Jop Zitman	2670863	j.zitman@student.vu.nl
Tiberiu Iancu	2659445	t.iancu@student.vu.nl
Sebastian Cristian Iozu	2663383	s.iozu@student.vu.nl
Yingdi Xie	2669853	y7.xie@student.vu.nl

Introduction

Author(s): Sebastian Cristian Iozu, Yingdi Xie

"Breach protocol" is part of the video game "Cyberpunk 2077"[1] developed by CD Projekt. In this game, the player attempts to solve a puzzle within a time limit. Specifically, the player is presented with a grid of cells, each cell consisting of 2 alphanumeric characters, along with a set of sequences of cells to be completed. The player picks cells to match the sequence and receives rewards corresponding to the matched sequence. There is a constraint on the player's moves that the player can only pick a cell in the same row or column as the last chosen cell except for the first move. There is a time limit indicated by the timer and also a limit to the number of cells to be picked indicated by the buffer size.

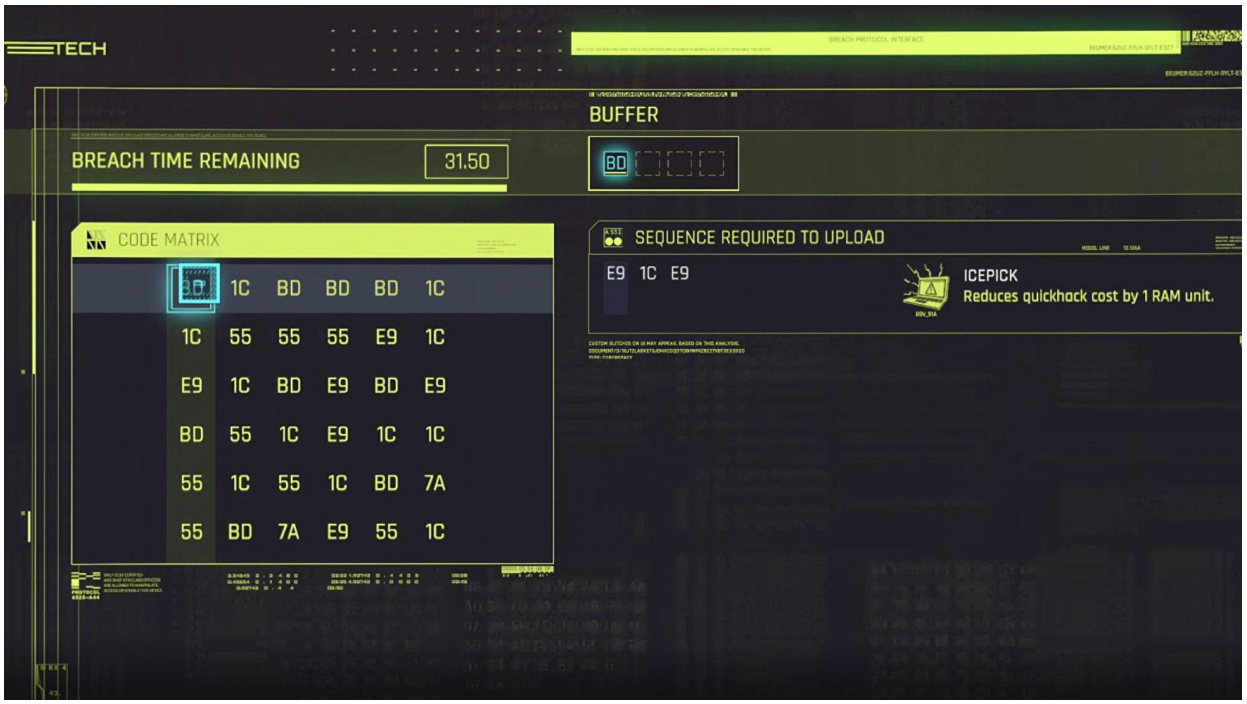

Figure 1. GUI of the original "Breach protocol"[2]

[1] www.cyberpunk.net/en/
[2] https://assets2.rockpapershotgun.com/cyberpunk-2077-hacking-minigame.jpg/BROK/resize/1760%3E/format/jpg/quality/80/cyberpunk-2077-hacking-minigame.jpg

Comparison with original version

For our project, we will develop a customized version of “Breach protocol”. We will preserve the main game logic as introduced above and the general setup of the original version. Meanwhile, because our game is independent from Cyberpunk 2077 and we intend to implement new functionalities, there are differences between our version and the original game. Below follows an overview of the commonalities and differences.

Commonalities

Our game preserves the GUI elements of the original game such as the grid of cells, the sequences to be matched and the buffer. Our game also displays a timer, which helps the user keep track of remaining time and makes the game more entertaining.

Similar to the original version, our game also provides immediate visual feedback to the user’s actions. For example, it shows a hovering effect when the user moves to a cell before confirming the selection. It highlights the sequences that the user has matched and fails to match with distinct background colors. This sort of feedback can guide the user while playing and enhances the game’s interactivity and responsiveness.

Differences

A major difference is that our implementation is not encapsulated within a game with larger goals. For example, the original game provides in-game rewards on completion of the minigame while our game stands alone and there is no larger goal that the user attempts to achieve. Furthermore, the original minigame is bootstrapped by Cyberpunk 2077, while our implementation requires manual ‘booting’.

Taking this difference into account, the minigame expects the user to manually specify a level text file while booting the game. The minigame will then load the level by reading that text file and parsing it into the game setup. We will ship an initial set of puzzles[3] for the time being, but our game is open to unlimitedly possible levels by adding and loading new text files. In addition, we may add a GUI level picker in a future update so that the user can more easily choose the level they want to play.

Our game includes functionality for a scoring system, which is not found in the original Cyberpunk minigame. The score is calculated using the following formula:

$$completedSeq! \ * \ timeLeft \ * \frac{completedSeq * (freeBufferCells + 1) * bufferSize}{100}$$

Our game allows the user to undo and redo their selection. While playing, the user may change their minds, discover a better strategy or intend to fix their mistakes. Our game allows the user to advance and reverse the game state and makes the playing process more dynamic.

Main audience

The main audience of our game is puzzle lovers. There is no specific limit on the age of the user, hence it suits both young and old adults. Anyone who has basic knowledge of using a computer and interacting with it via a keyboard should be able to play our game. It can run as a stand-alone game which helps those who feel bored in difficult times such as during corona lockdown and need to kill time. By separating this game from Cyberpunk 2077, we strive to make it available for a broader audience.

---

[3] https://github.com/kyle-rader/breach/tree/main/puzzles/txt

User scenario

As soon as the user starts the game, a level will be loaded and displayed on the screen. The timer will start counting down as soon as the user selects their first cell. Using the arrow keys, the user can navigate through the matrix and select a character using the `Spacebar` key. In case a mistake was made, the `Backspace` button can undo the selected characters one by one. If the wrong character was deleted, it can be brought back by pressing the `R` key. When the player is finished, they can simply press `Enter` or wait for the timer to reach 0, at which point their score and time left will be displayed.

Features

Functional features

Author(s): Jop Zitman

In the following table, we thoroughly describe each feature of the minigame to be implemented. The features are separated in such a way that they can easily be implemented by a single champion. We hope to be able to modularize as many features as possible to improve maintainability. Furthermore, the planned order of implementation mostly corresponds to the order of occurrence in the table.

ID	Short name	Description	Champion
F1	Load level	Read level from text file (path given as an argument to the jar) into Java variables (i.e. buffer size, matrix/grid values, and target sequences).	Chris
F2	GUI	Using libGDX[4], open a new window and draw a basic buffer, matrix/grid, target sequences and a timer to mimic the original game.	Chris
F3	Exit	The user can exit the game at any time. Any progress will be lost.	Chris
F4	Move selector	The user can move their current selector horizontally/vertically inside the 5x5 matrix.	Yingdi
F5	Confirm selector	The user can confirm their current selector. The value of the selected cell is added to the buffer and the UI is updated.	Tiberiu
F6	Visual feedback	The game shows a hovering effect over the currently selected cell and corresponding rows and columns. It also highlights the sequences that the user has matched and fails to match with distinct background colors.	Yingdi
F7	Undo select	The user can undo their last step. The last value is the buffer is removed and the UI is updated.	Jop
F8	Redo select	The user can reverse their previous undo step. The undone selection is taken again and the UI is updated.	Jop
F9	Game end	The user's score is calculated[5] and presented together with the corresponding buffer and sequences.	Yingdi

[4] See the libraries section

[5] See the score calculation in the differences section

<table>
<tr><td>F10</td><td>Timer</td><td>The timer starts at a fixed time and only starts counting down when the user has made its first move. Once it has reached zero, the user can no longer make moves and the game ends (F9).</td><td>Tiberiu</td></tr>
<tr><td>F11</td><td>Confirm buffer</td><td>The user can confirm their current buffer and stop the timer. The game ends (F9).</td><td>Tiberiu</td></tr>
<tr><td>F12</td><td>Input handling</td><td>Only keyboard input is handled. The following keys are mapped:<table><tr><td><code>Escape</code></td><td>Exit game (F3)</td></tr><tr><td><code>Up</code>/<code>Down</code>/<code>Left</code>/<code>Right</code></td><td>Move selector (F4)</td></tr><tr><td><code>Spacebar</code></td><td>Confirm selector (F5)</td></tr><tr><td><code>Backspace</code></td><td>Undo select (F7)</td></tr><tr><td><code>R</code></td><td>Redo select (F8)</td></tr><tr><td><code>Enter</code></td><td>Confirm buffer (F11)</td></tr></table></td><td>Jop</td></tr>
</table>

Quality requirements

Author(s): Tiberiu Iancu

ID	Short name	Quality attribute	Description
QR1	Input validation	Reliability	When the player presses a key, the input handler should recognize if the move is illegal and prevent any game crash/error that might come because of it. This also includes illegal undoing/redoing of moves.
QR2	Unlimited number of possible levels	Maintainability	New levels could be added to the game as text files, and loaded through F1.
QR3	Valid levels	Reliability	The application should only provide solvable puzzles to the user: the puzzles themselves should be tested before being included in the game and the level loader should be able to correctly parse the text files.
QR4	Procedural level generation can be added in an update	Maintainability	The app should be designed to allow for procedural level generation in the future.
QR5	Level picker can be added in an update	Maintainability	The app should allow for future addition of a level picker.
QR6	Platform independence	Reliability	The score system and the timer should behave the same regardless of the platform the app is running on.

QR7	Common input method	Usability	The app should use an input method common in video games, that the user can easily get accustomed to.
QR8	System safety	Security	The app shouldn't be able to compromise user data, corrupt memory etc. The app shouldn't be able to access user files outside its installation directory. The app should also be unable to connect to the Internet for increased privacy.
QR9	Corrupted JAR protection	Security	Alongside the app, we will also publish the checksum of the JAR. This can assure the user that the application hasn't been modified by a malicious user.

Java libraries

Author(s): Jop Zitman, Tiberiu Iancu

libGDX
Used for creating the GUI of the game. We chose it among others because libGDX provides suitable utilities for simple 2D game development such as straightforward UI elements and input handling. It is also well documented and thus it is rather easy to find answers in case of doubts. We also see games developed with libGDX presenting similar GUI as our design.

Lombok
Used for easy getter/setter creation for cleaner code. We chose it because it automatically generates common methods such as getter and setter in the bytecode, thus speeding up the implementation and improving the maintainability of our code.

JUnit 5
Used for unit testing code features. With unit testing, we hope to improve both maintainability and reliability of the code. Unit testing forces the developer to think of breaking edge cases and often helps keeping the code modular.

Time logs

Team number	14		
Member	**Activity**	**Week number**	**Hours**
Jop	Create Github repo	1	0.5
Everyone	Group meeting (general discussions)	1	1
Everyone	Group meeting with TA	1	0.5
Jop	Written FF	1	1
Tiberiu	Written QR & added some libraries	1	1
Chris	UI Library hunting	1	1
Jop	Feedback QR	1	0.25
Everyone	**Heated** argument about FF's	1	0.5
Yingdi	Library hunting and writing intro	1	0.5
Tiberiu	Finished QR	2	1
Chris	Written (part of) Intro + comments	2	1
Jop	Feedback intro & improve FF	2	1
Everyone	Even a more **heated** argument about scoring system	2	0.5
Yingdi	Written (part of) Intro + comments	2	1
Everyone	Review of the document	2	0.5
		TOTAL	11.25