

The Great Escape Of Julia Gadget

User Manual

Menu

After you have launched the game (see Installation Guide), you will immediately be faced with the main menu, from which you have six options to select from.

- 'Start', will begin the game.
- 'Quit' will exit the game.

How To Play

As you attempt to make your way across the university campus to your car, you will pass through different departments and find yourself besieged by various students with their queries. Some, but not all, of these queries will be appropriate for the present department, though some will be more general. Upon being asked a question, you will be given several choices of response. Some of your choices of answer will be positive and helpful for the student, whilst some will be negative and unhelpful. You must answer five questions in order to exit the department, and pass through five departments to reach your car.

Controls

Enter/Left Click - Advances dialogue and activates dialogue
Space - Advances dialogue (without selecting choices)
Arrow Keys - Navigate the interface
Escape/Right Click - Accesses the game menu
Ctrl - Skips dialogue while held down
Tab - Toggles dialogue skipping
Page Up/Mouse Wheel Up/Click Rollback Side - Rolls back to earlier dialogue
Page Down/Mouse Wheel Down - Rolls forward to later dialogue
H/Middle Click - Hides the user interface
S - Takes a screenshot
Shift+A - Opens the accessibility menu

Game Menu

Upon accessing the game menu, you will have the following options to choose from:

- 'History', which will show a transcript of game text and dialogue up to this point.
- 'Save' will save the game.
- 'Load' will load a saved game.
- 'Preferences', 'About', 'Help' and 'Quit', which do the same as their respective namesakes in the main menu.
- 'Main Menu', which will take you to the main menu WITHOUT saving.
- 'Return' will exit the game menu and return you to the game.

Note that some of these options are available at the bottom of the interface without having to access the game menu.

The Map

While between departments, you will access the map of the university. The departments you must travel through are labeled clearly. You can choose to enter the departments in any order, but

must select each one once in order to traverse the university and win the game. To select a department and enter it, simply click on the label, or navigate through the labels using the arrow keys and pressing enter once the intended department is selected.

Stats

Julia Gadget has three stats; logic, creativity and debating. Whilst interacting with students, you can view Julia's stats by selecting the statistics symbol in the top right of the interface, and similarly exiting this menu by selecting 'Return' in the top right. At the beginning of the game, Julia will have a value of 1 for each of these stats. In dialogue, some of Julia's responses will require a certain amount of logic, creativity or debating skill. If you select a response that you do not have the sufficient statistical trait for, you will be informed as such and asked the question again.

University Ranking

As you interact with students, your responses will have consequences. The university ranking, visible in the Ranking Meter in the top left of the interface throughout dialogue, will be directly affected by your decisions. If you provide a particularly helpful and/or positive response to a student, the university ranking is liable to increase. Similarly a rude, offensive or unhelpful response is liable to cause the university ranking to decrease. Upon completing the game, the ranking, and by extension your choices, will affect the outcome of Julia's cross-campus odyssey.

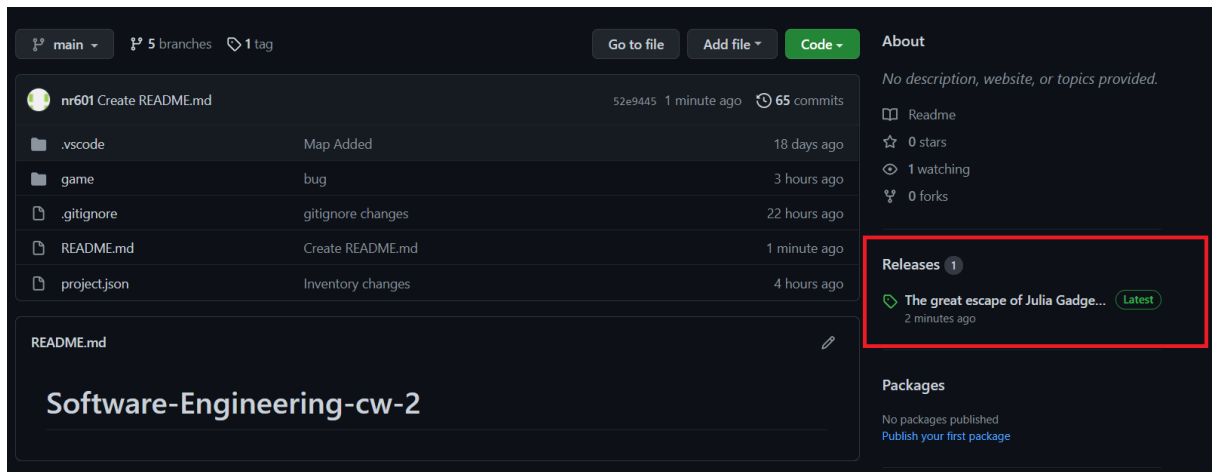
Leaderboards

After completing the game, you will be shown the leaderboard, where you can compare your score (final university ranking) with that of all your friends, who also enjoy a good game of 'Julia Gadget'. At your own discretion, you can select 'Add to Leaderboard', then add your own name to supplement your score, thereby rubbing it in the faces of all your chums with lower scores. You can also choose to 'Clear Leaderboard', likely out of petty jealousy and spite directed at the skilled individuals who sit at the top of the leaderboard. Finally, you can also 'Exit Game' if you've just had enough.

There are two ways for running this game, one is directly through the executable file, and the other one is through source code.

1) Running through executable file





- Go to [markas9/Software-Engineering-cw-2 at delivery \(github.com\)](https://github.com/markas9/Software-Engineering-cw-2) to access the 'delivery' branch of our project repository.
- Open the project release from the repository.



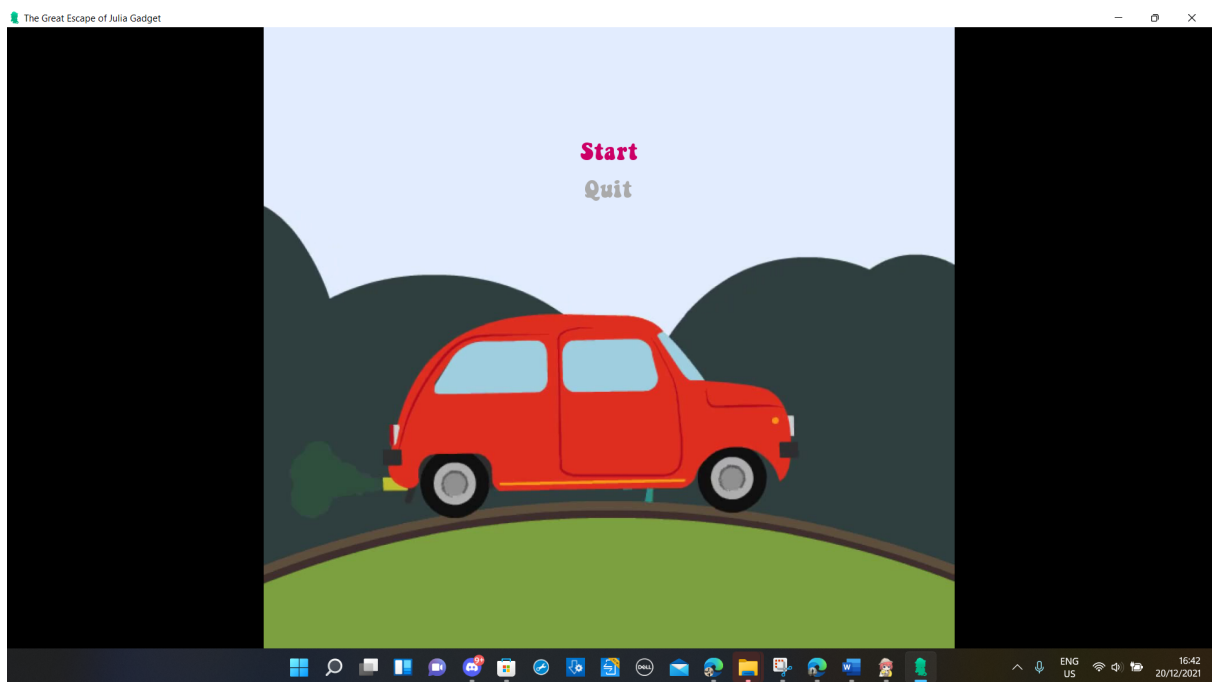
- Then download the zip file from the release based on your operating system, mac for Macintosh and pc for Windows and Linux.



- After downloading the zip file, extract it's content and run the application file from the extracted folder.

game	20/12/2021 15:38	File folder	
lib	24/10/2021 01:57	File folder	
renpy	23/11/2021 00:20	File folder	
 TheGreatEscapeofJuliaGadget.exe	24/10/2021 01:57	Application	121 KB
 TheGreatEscapeofJuliaGadget.py	24/10/2021 01:57	Python Source File	7 KB
 TheGreatEscapeofJuliaGadget.sh	24/10/2021 01:57	Shell Script	2 KB
 TheGreatEscapeofJuliaGadget-32.exe	24/10/2021 01:57	Application	115 KB

- Click on “Start” to launch the game.

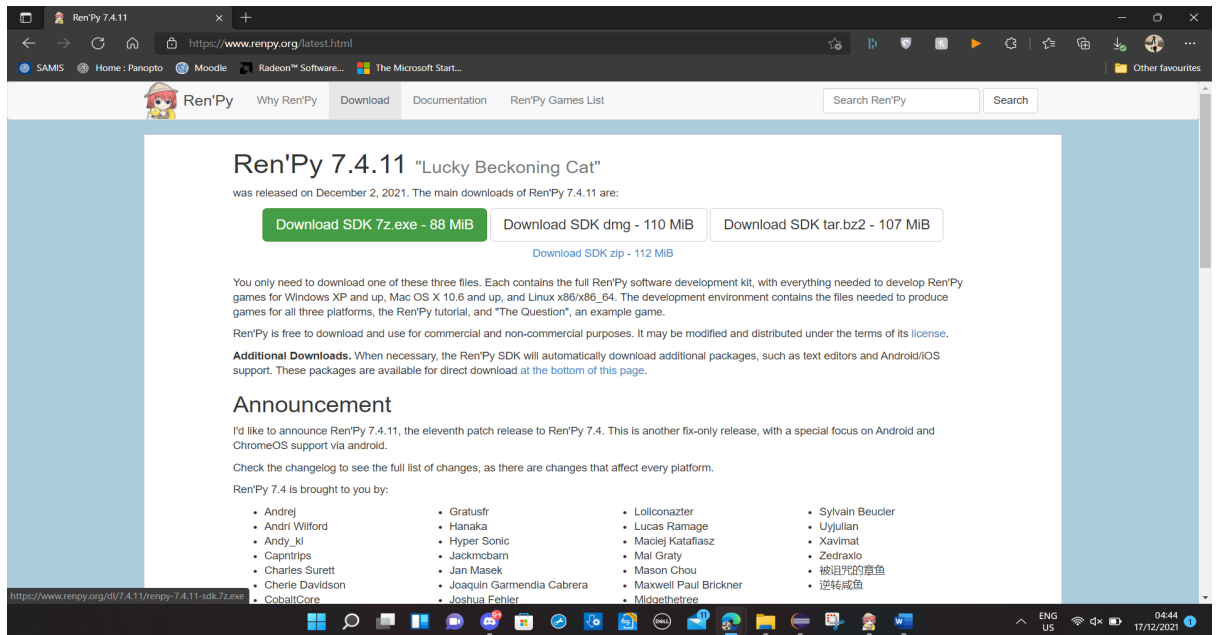


2) Running through source code

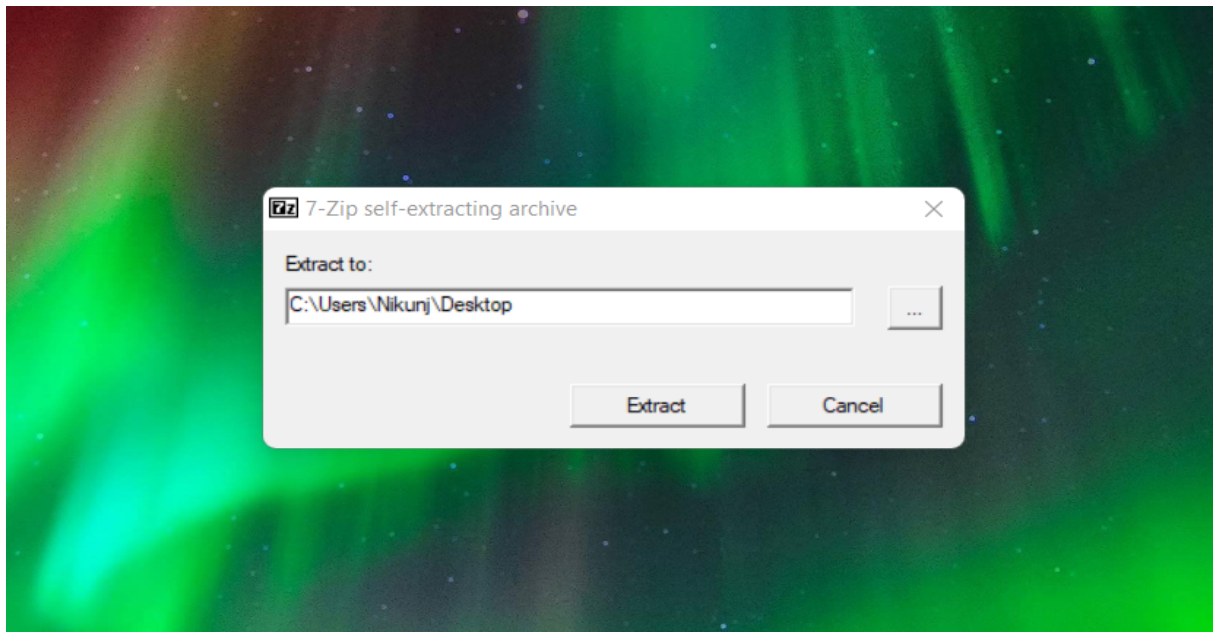
For running the game through source code, we first need to install Renpy and then clone our git repository in the Renpy folder.

2.1) Renpy Installation

- Go to [Ren'Py 7.4.11 \(renpy.org\)](https://www.renpy.org) and download executable file (.exe) for renpy.



- Run the executable file.
- Select the location where you wish to install renpy in your system.

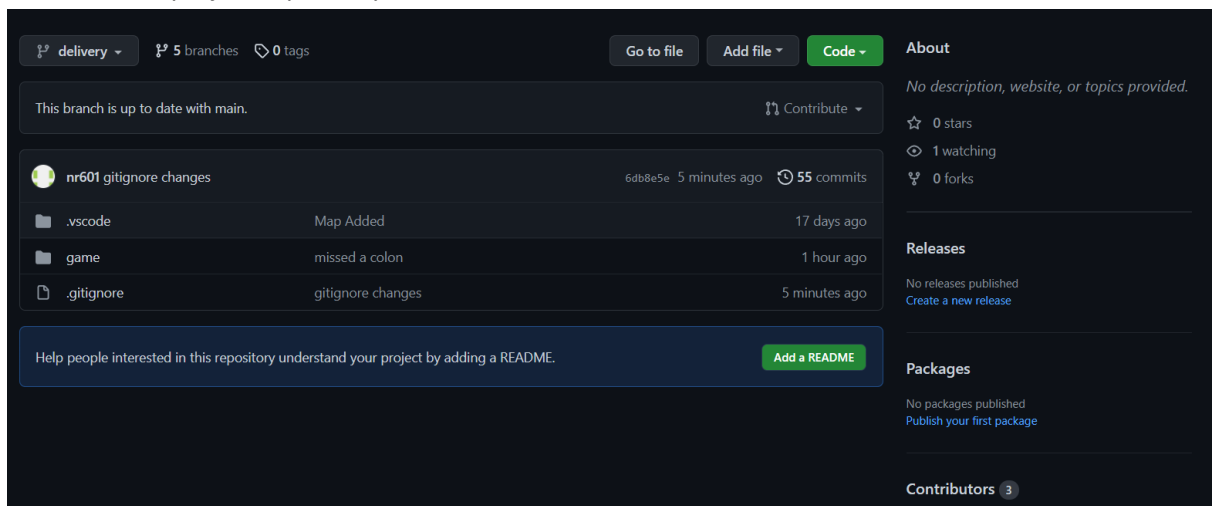


This will create a Renpy folder on your desktop (or at your preferred location for installation).

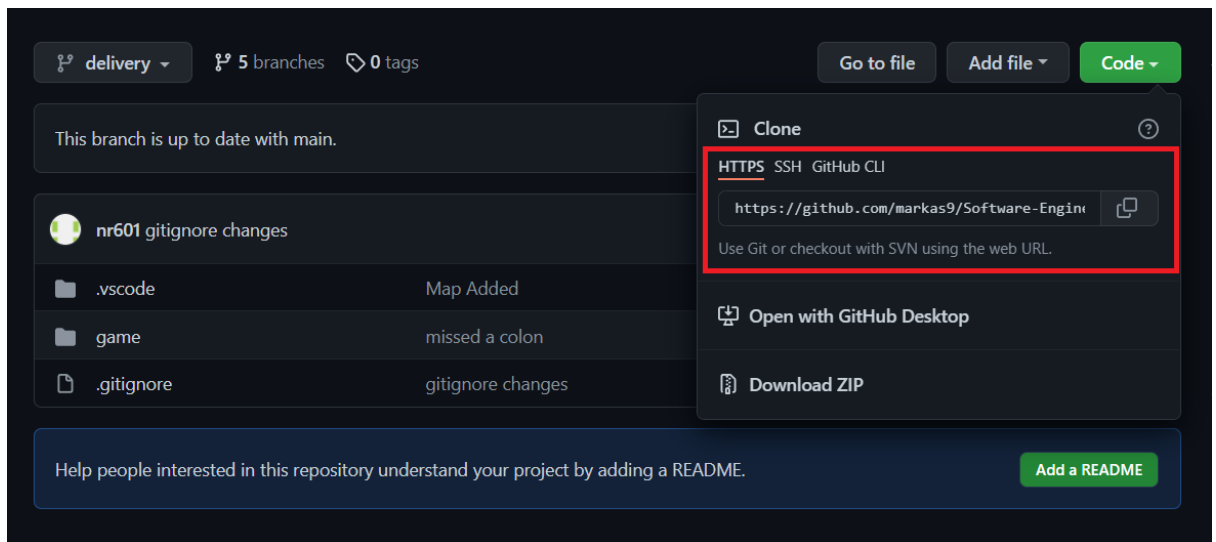
- After the installation is completed, you can find Renpy within system apps.

2.2) Cloning Git project onto your system

- Go to [markas9/Software-Engineering-cw-2 at delivery \(github.com\)](https://github.com/markas9/Software-Engineering-cw-2-at-delivery) to access the 'delivery' branch of our project repository.



- Copy the Https clone link for the repository.



- Open your command line and redirect to your Renpy folder

```
Command Prompt
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nikunj>cd Desktop
C:\Users\Nikunj\Desktop>cd RenPy
C:\Users\Nikunj\Desktop\RenPy>
```

- Enter git command for cloning the project, *git clone [https link you copied](https://github.com/markas9/Software-Engineering-cw-2.git)*

“git clone <https://github.com/markas9/Software-Engineering-cw-2.git>”

```
Command Prompt
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nikunj>cd Desktop
C:\Users\Nikunj\Desktop>cd RenPy
C:\Users\Nikunj\Desktop\RenPy>git clone https://github.com/markas9/Software-Engineering-cw-2.git
Cloning into 'Software-Engineering-cw-2'...
remote: Enumerating objects: 971, done.
remote: Counting objects: 100% (218/218), done.
remote: Compressing objects: 100% (181/181), done.
remote: Total 971 (delta 53), reused 182 (delta 37), pack-reused 753 receiving objects: 100% (971/971), 69.13 MiB | 7.84 MiB/s
Receiving objects: 100% (971/971), 72.73 MiB | 7.33 MiB/s, done.
Resolving deltas: 100% (284/284), done.

C:\Users\Nikunj\Desktop\RenPy>
```

- After that change directory to the clone repository, then run *git checkout delivery* to switch to delivery branch.

“git checkout delivery”

```
Command Prompt

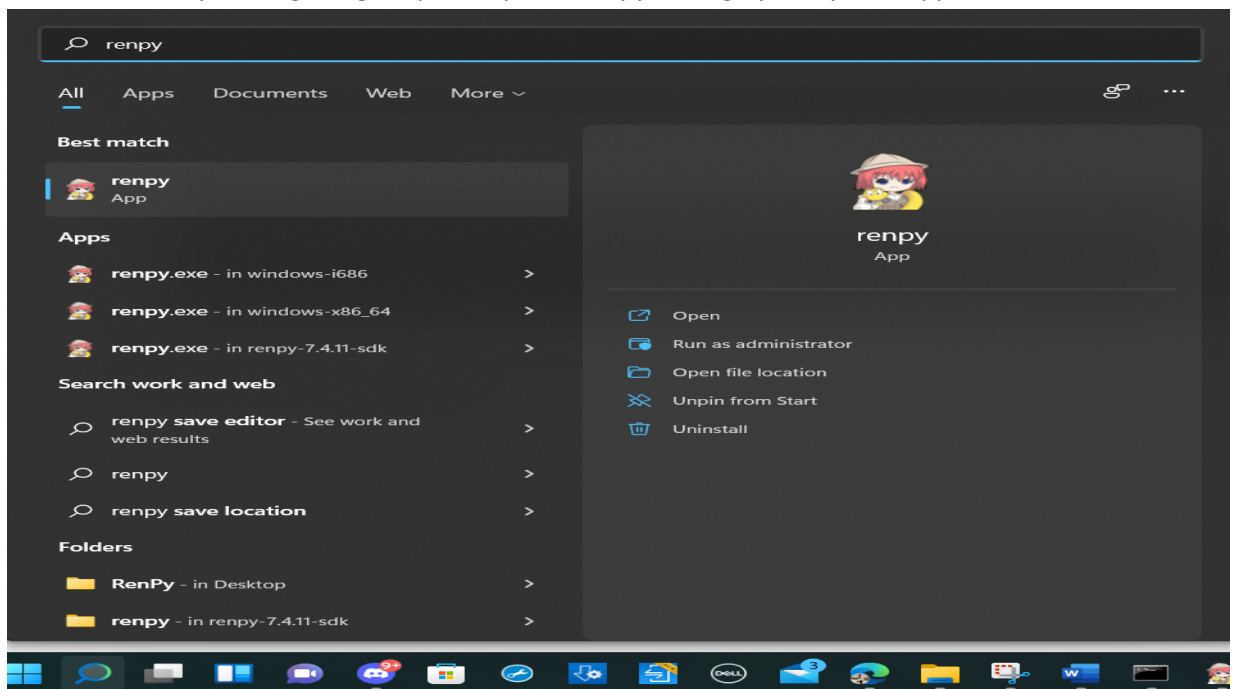
C:\Users\Nikunj\Desktop\RenPy>cd Software-Engineering-cw-2

C:\Users\Nikunj\Desktop\RenPy\Software-Engineering-cw-2>git checkout delivery
Switched to a new branch 'delivery'
Branch 'delivery' set up to track remote branch 'delivery' from 'origin'.

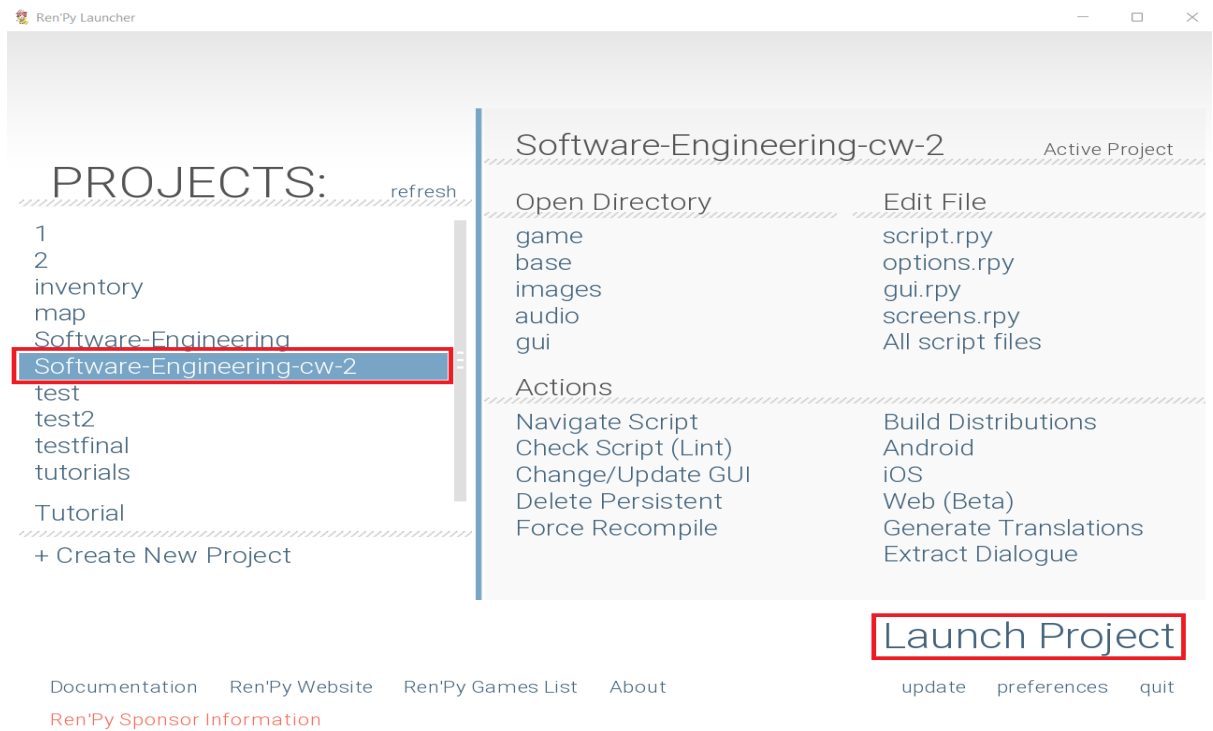
C:\Users\Nikunj\Desktop\RenPy\Software-Engineering-cw-2>
```

2.3) Running the game through Renpy

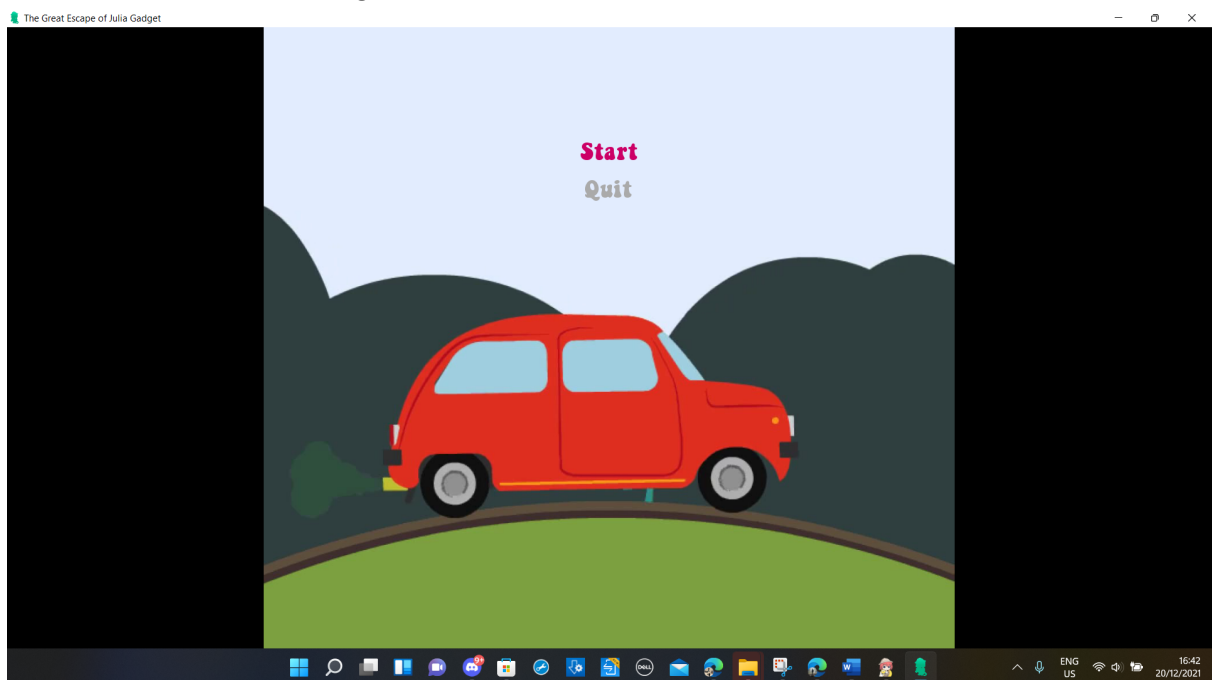
- After successfully cloning the git repository, run Renpy through your system apps.



- After running the Renpy software, launch “Software-Engineering-CW-2” project through Renpy.



- Click on “Start” to launch the game



Maintenance Guide

Navigating Ren'py code

To access renpy code from the launcher (see installation guide) select the project "The Great Escape of Julia Gadget" and then click on "Open Project" under the category "Edit File". If this is the first time doing so, Ren'Py will ask you to select a file editor such as *Atom*, *VStudio*, *VIM*, etc. After the file editor opens, you can click on any of the files listed in the Maintenance Guide to make alterations to the game.

1 - script.rpy

This file contains "*label start*" and it is a special label because it is where Ren'py will start its script when the user clicks "Start Game" from the main menu. This script file also contains the settings of the initial scene where the background of University of Lath is shown along with the narration for the purpose and aim of this game.

From here the game will jump to the map screen where the user will be able to choose one of 5 departments to go through and answer questions and be rewarded with University ranking.

2 - Map Folder

2.1 - mapscreen.rpy

2.2 - scripts.rpy

This folder contains all the files for Map. Within scripts.rpy there is the class (Place) for placing different sections of the game over the Map, if a new section is to be added or old one is to be deleted, it has to be done through here. And mapscreen.rpy is responsible for the UI of map and for presenting the data over the map.

3 - Departments Folder

3.1 - artdep.rpy

3.2 - csdep.rpy

3.3 - languagesdep.rpy

3.4 - mathdep.rpy

3.5 - musicdep.rpy

All department files are functionally similar and contain a label which defines a player's run through each department. They are separated into their own files such that they can be further

adapted to contain elements unique to each department, but currently they differ only in a few key ways:

- The images, specific to the student within this department, using the format of a normal pose, a smiling pose, and a frowning pose. Character images are contained in the folder `game/images/`
- The scene, which is an image contained within `game/images/rooms` to display as the background of the department.
- The variables utilized
 - `'{department}_dep_visited'` is set to true, to indicate the department has been visited
 - `'rankingBefore'` is set to the ranking of the player as they enter the department, such that it can be compared to their ranking on exit
 - A student object, `'{department}_Student'` is created and used to obtain the questions asked within the department, via its `getQuestion` Function.

Then at the end of each department the notifications from `phone_notifications` are called. If one wishes to create new departments, they should just copy a prior department, adjust the variables and mentioned differences, then include it within the map in the Map Folder.

4 - Questions folder

- 4.1 - `artQuestions.rpy`
- 4.2 - `csQuestions.rpy`
- 4.3 - `genericQuestions.rpy`
- 4.4 - `languagesQuestions.rpy`
- 4.5 - `mathQuestions.rpy`
- 4.6 - `musicQuestions.rpy`
- 4.7 - `student.rpy`

Within each `{department}Questions.rpy` file there are each of the questions that students could ask along with a list grouping that type. To create new questions the format of the current questions should be followed, and the question label should be added to the appropriate list at the top of each file.

`student.rpy` contains the student object. This object allows for questions from the question lists to be randomly selected and presented to the user, via the `getQuestion` and `getQuestionType` functions. `getQuestionType` returns an index based upon the probabilities defined at object initialisation, this index is then used within `getQuestion` to pop a question (such that it is not asked again) from the respective question list and return it. Currently it is only implemented to work with two lists. One list of passed questions, the specific department questions, and the generic question list. However the `getQuestion` function can be easily adapted to accept multiple

question lists, however the probabilities variable will have to be extended also to contain the same number of elements as the number of question lists you wish to be chosen from.

5 - functions.rpy

This script file contains the initialization of the game along with all the functions that are defined in python. Check the table below to see the custom defined functions along with their description.

Function	Description
askUsername()	This function is called when the user decides to add his score on to the leaderboard. The user inputs their username which is displayed on the board along with their score.
addToLeaderboard(username)	This function takes as a parameter the username of the player and stores it in the game's persistent data. These are data which are stored and remain throughout different runs of the game. The function only adds up to the top 10 scores inputted into the scoreboard and displays them from the highest to the lowest score.
change_music()	This function is called whenever the player reaches the map stage and checks their University ranking score. The music changes according to how high it is.
check_ending_scene()	This function will display at the end of the game a different background which will showcase how well the player did according to how high their University ranking score is. The higher the score is, the better condition Julia's car will be.
all_visited()	This function checks whether all the departments have been visited and then if that's true then it changes the stage of the game where Julia reaches her car.

6 - phone_notifications.rpy

This script file contains a screen that is displayed at the end of every department. The purpose of this screen is to simulate the player receiving a text message that tells them their progress in the game by what their current University ranking is.

7 - parkinglot.rpy

This script file contains the narration and events that take place at the last stage of the game, the parking lot. At this stage the player can no longer answer questions and change their University ranking. The ending that they get is dependent on how high they scored, which changes the background they see with the narration they are given.

Additionally at the bottom of this file is also where the leaderboard is called.

8 - leaderboard.rpy

This file contains the screens that are displayed and make up the leaderboard. Firstly there's the *scoreboard* which showcases the user's top 10 University ranking scores. Then there are the buttons that can interact with the leaderboard which are the screens `add_to_leaderboard_button`, `clear_scoreboard_button` and `exit_game_button`. Each one has an idle and hover display and all of them jump to a different label that is placed at the bottom of the file. The first button allows the user to add to the leaderboard but only once per run and only if they have scored high enough. The second button clears all the data currently listed in the leaderboard and the last button simply Exits the game.

9 - Audio folder

This folder contains all the audio that is used inside the game.

10 - Images Folder

This folder contains all the images and symbols that are used inside the game.

11 - options.rpy

This renpy file allows for changing options

See here for more information

<https://videlais.com/2018/07/16/customizing-renpy-part-1-editing-options-rpy/>

12 - screens.rpy

This renpy file determines the layout for screens used in the project.

13 - gui.rpy

This renpy file is used for Graphical user interface customization.

See here for more information

[Customizing Ren'Py: Part 2: Editing gui.rpy – Digital Ephemera \(videlais.com\)](#)

14 - Inventory Folder

- 14.1 - inventory.rpy
- 14.2 - inventoryFeatures.rpy
- 14.3 - inventoryUI.rpy
- 14.4 - itemList.rpy

Then for the future scope of our game we have added inventory to our game. It does not currently fit in with the storyline and other features of the game, but we have included it in respect with the future scope of our game.

inventory.rpy holds the section for accessing inventory and market.

ItemList.rpy holds the item that is represented in inventory and market.

inventoryUI.rpy is responsible for the overall UI of the inventory and market.

inventoryFeatures is responsible for major inventory and market functionality.

15 - Extending code for further development

The code can be altered to make changes to the game however one wants to, but the changes must be done in the scriptive language of Ren'py or python that is placed inside of python blocks. All Ren'py scripting files must have an extension that ends in *.rpy*.