Ripple Effect Guide

Compiled by
Ryan El Khoury, Romy Nissan

Manual Guide

Credit

Ryan El Khoury - 2230344
Romy Nissan - 2230908
Dimitri Perron - 2230233
Zayden Kung’u - 2232072

Overview

Introducing "Ripple Effect," an interactive storytelling project crafted with Python and Tkinter. Inspired by the immersive gameplay of Lifeline, this project invites users to explore dynamic narratives through a graphical user interface (GUI).

With "Ripple Effect," users navigate branching paths and make pivotal choices that shape the direction of the story. You're in charge. Your decisions shape the story. It's all about exploring different paths and seeing where your choices take you.

When the game starts, pick a character and choose one of three built-in stories, or make and implement your own! When the game starts, you’ll be put into a situation based on your story choice and you can choose what your character should do. However, different actions have different consequences. In some endings you live, some you die, and some you get lost forever.

Installation and Instructions

The game was built using Python 3.11 and the modules used were Tkinter, os, playsound, and PyGames.

Installation:
Make sure all the modules are installed. If using Python 3.12 or higher, there may be an issue with playsound. It has not been tested. The safest case would be to run Python 3.11.

Instructions:

1. Run the ui.py file to start the game.
2. Press start game.
3. Pick your character! This choice will not affect any outcomes.
4. You’ll be presented with three different stories:
    a. Spacecapade: You are a NASA astronaut on an unknown planet and find out that your spaceship is broken. It’s your job to figure out how to fix your ship and save yourself.
    b. Forestrealm: You wake up in a forest with no clue where you are. It is your job to save yourself. Watch out for any dangers lurking in the forest with you
    c. Home Alone: You are in your room, trying to pass the time. You receive an alert from your security system and suspect that someone broke into your house. It is your job to investigate and find out what is going on.
    d. An option to enter your own. Enter the files in their respective designations. If an example is needed, enter *script0.txt* and *edgerep0.txt*.

5. Each story has multiple endings, so they can each be replayed multiple times with different outcomes.
6. If you wish to implement your own script, you must follow a specific format for your script file and edge file. These text files MUST be put in the scripts folder:
    a. Script file: It must follow the same punctuations and format as shown below. If you wish to add any comments, make sure it starts with a hashtag.

        Question Number
        Question:*Left or right?*
        Answer1:*Right*
        Answer2:*Left*

    b. Edge file: The numbers must follow the same punctuation with no whitespace at the ends. BE SURE to map the number answer 1 corresponds to on top of answer 2's number.

        1 2
        1 3
        2 4
        2 7

7. For quick ways to restart or quit the game, press "q" on your keyboard to quit, and "r" to restart.
8. There is a small easter egg in the game just for you sir. Enter your own script, and enter the keyword *vinnie* and make sure your volume is all the way up!

Design Guide

Organization

The software organization was done as follows:

1. The backend file *sourcecode.py* file that implements the digraph and story class was coded by Ryan.
2. The front end file *ui.py* file that codes the animations was created by Romy.
3. The scripts found in the *scripts* folder that are implemented in the game were written by Dimitri.
4. The *README.txt* was written by Zayden.

The folder organization:

1. The folder “scripts” contains all the scripts read by our program.
2. The folder “bg” contains all the images in the UI file.
3. The folder “sounds” contains all the background sounds and sound effects throughout the game.

Source Code File (*sourcecode.py*)

The source code file was implemented using two main classes:

digraph Class:

- Handles the construction and traversal of the game's stories.
- Manages the adjacency list for storing page connections.
- Implements methods for adding edges, reading edge files, constructing a choices dictionary from a script file, and accessing the next page and question.

story Class:

- Inherits from the digraph class.
- Manages the overall flow of the game.
- Handles user input, updates the current page based on player choices, and retrieves the corresponding question and answers.
- Provides methods for starting the game, handling left and right button clicks, and checking script and edge file legality.

User Interface File (*ui.py*)

The UI file was implemented using six classes to represent each page of the game:

1. HomePage Class:
   - Represents the start menu with a title, play game, and exit game button.
2. GamePage Class:
   - Represents the character selection page with four different character button options.
3. SettingPage Class:
   - Represents the scripts page, which offers the choice of three different built-in script buttons and the option to import your own.
4. PlayPage Class:

        - Represents the main playing page.
        - Uses a countdown to start the game and then follows by displaying the story and having the buttons and questions reconfigure as the story develops.
5. SpecialPage Class:
    - Represents the "just for fun" page. A fun way to say thank you for the past year!
6. Main Class:
    - Represents the main application window.
    - Implements methods for switching between different pages, handling keyboard events for fast quit and restart.

Techniques

1. Object-Oriented Programming (OOP):
    - Used in both files.
    - Classes are used to represent the different pages and or different methods implemented.
2. File Handling:
    - Input/output operations are performed to read script and edge files.
3. Error Handling:
    - Exception handling is implemented to deal with potential errors, such as files not found or invalid inputs.
4. Data Structures:
    - The digraph data structure (implemented as an adjacency list) is used to represent the story's narrative structure.
5. Algorithm Design:
    - Graph traversal algorithms are employed to navigate through the story pages based on the player's choices.
6. Event Handling:
    - Event bindings are used to trigger functions in response to user interactions such as button clicks and keyboard events.

Sources

The main sources used:

- Stackoverflow
- Github
- GeeksforGeeks
- PythonDocs
- Youtube

Tools

- Python: The source code is implemented in Python.
- Tkinter Library: Utilized for GUI development.
- Pygame Library: Used for playing sound effects.