Bìa

* Thành viên nhóm
* Tên trường, tên project, tên lớp

Giới thiệu

Lời cảm ơn

Table of Contents

[**1.** **Introduction to Game RushRelic (mordern remake of RushHour)** 5](#_Toc202695133)

[**a.** **Installation** 5](#_Toc202695134)

[**i.** **Github** 5](#_Toc202695135)

[**ii.** **Zipped file** 5](#_Toc202695136)

[**b.** **Interface** 5](#_Toc202695137)

[**i.** **Screens layout and flow** 5](#_Toc202695138)

[**ii.** **Gameplays** 5](#_Toc202695139)

[**c.** **Design system and Design pattern** 5](#_Toc202695140)

[**i.** **System Class** 5](#_Toc202695141)

[**ii.** **Class relative to Game: Map and Vehicle** 5](#_Toc202695142)

[**iii.** **Constants** 5](#_Toc202695143)

[**2.** **Description of the Maps** 5](#_Toc202695144)

[**a.** **Foudational Elements in a Map** 5](#_Toc202695145)

[**b.** **Map Details** 5](#_Toc202695146)

[**i.** **Map 1** 5](#_Toc202695147)

[**ii.** **Map 2** 5](#_Toc202695148)

[**iii.** **Map 3** 5](#_Toc202695149)

[**iv.** **Map 4** 5](#_Toc202695150)

[**v.** **Map 5** 5](#_Toc202695151)

[**vi.** **Map 6** 5](#_Toc202695152)

[**vii.** **Map 7** 5](#_Toc202695153)

[**viii.** **Map 8** 5](#_Toc202695154)

[**ix.** **Map 9** 5](#_Toc202695155)

[**x.** **Map 10** 5](#_Toc202695156)

[**3.** **Search Algorithm Description** 6](#_Toc202695157)

[**a.** **Input and Output Specification** 6](#_Toc202695158)

[**b.** **State Representation and Visited Table** 6](#_Toc202695159)

[**c.** **Solvers explaination** 6](#_Toc202695160)

[**i.** **BFS Solver** 6](#_Toc202695161)

[**ii.** **DFS Solver** 6](#_Toc202695162)

[**iii.** **UCS Solver** 6](#_Toc202695163)

[**iv.** **A\* Solver** 6](#_Toc202695164)

[**4.** **Experiments** 6](#_Toc202695165)

[**a.** **Search Time** 6](#_Toc202695166)

[**i.** **Measurements** 6](#_Toc202695167)

[**ii.** **Visualizations** 6](#_Toc202695168)

[**iii.** **Insight** 6](#_Toc202695169)

[**b.** **Memory Usage** 6](#_Toc202695170)

[**i.** **Measurements** 6](#_Toc202695171)

[**ii.** **Visualizations** 6](#_Toc202695172)

[**iii.** **Insight** 6](#_Toc202695173)

[**c.** **Expanoded** 6](#_Toc202695174)

[**i.** **Measurements** 6](#_Toc202695175)

[**ii.** **Visualizations** 6](#_Toc202695176)

[**iii.** **Insight** 6](#_Toc202695177)

[**5.** **Project Plan and Task Distribution** 6](#_Toc202695178)

[**a.** **Project Plan** 6](#_Toc202695179)

[**b.** **Task Distribution** 6](#_Toc202695180)

[**6.** **References** 6](#_Toc202695181)

[**7.** **Apendix** 6](#_Toc202695182)

1. **Introduction to Game RushRelic (mordern remake of RushHour)** 
   1. **Installation** 
      1. **Github**
      2. **Zipped file**
      3. **Video Youtube**
   2. **Interface**
      1. **Screens layout and flow**
      2. **Gameplays**
   3. **Design system and Design pattern**
      1. **System Class** 
         1. Class Program

The game is built using **pygame**, a Python library for 2D game development that supports graphics rendering, user input, and sound.

The main function is the core of the system, structured as a continuous game loop. Each loop **clears the screen**, **processes input**, **updates object states**, and **redraws the frame**.

To ensure only one controller instance, the **Singleton** pattern is applied to main.

Rendering uses **double buffering**: objects are first drawn to a **back buffer**, then flipped to the screen for smooth display and reduced flickering.

* + - 1. Class Screen

This class applies the **State design pattern** to manage different UI screens. New screens can be added using the **addScreen(‘hello\_screen’)** method, and the current screen can be switched using **setScreen(‘hello\_screen’)**.

* + - 1. Class Graphic

This class also follows the **Singleton** design pattern. It centralizes all rendering-related functions using pygame, ensuring that the library is imported and used only once across the entire source code.

* + - 1. Class Solver

The Solver class is a core component responsible for implementing the game's solving algorithms. It includes a **BaseSolver** class that contains shared logic and helper functions used across all strategies, ensuring consistency and reducing code duplication.

To support scalability, a **SolverFactory** is used to create solver instances dynamically, allowing new algorithms to be added with minimal changes.

The system adopts the **Strategy design pattern**, enabling the program to switch between different solving algorithms (e.g., BFS, DFS) at runtime. Each strategy is encapsulated in a separate class, making the design modular, extensible, and easy to maintain.

* + - 1. Class relative to UI
      2. Class Audio
      3. Class Clocker
    1. **Class relative to Game: Map and Vehicle**

Phần này sẽ định nghĩa chi tiết về Class Vehicle và Map vì chúng ảnh hưởng trực tiếp đến các nội dung ở dưới, cụ thể:

* + - 1. Vehicle

Gồm 5 tham số đầu vào chính là:

* orient: hướng của xe
* len: độ dài của xe
* x, y: tọa độ của xe
* name: tên của xe
  + - 1. Map
    1. **Constants**

1. **Description of the Maps** 
   1. **Foudational Elements in a Map**

Each map in *RushRelic* is designed based on a consistent set of foundational elements that define the gameplay experience. These include:

* 1. **Map Details**
     1. **Map 1**
     2. **Map 2**
     3. **Map 3**
     4. **Map 4**
     5. **Map 5**
     6. **Map 6**
     7. **Map 7**
     8. **Map 8**
     9. **Map 9**
     10. **Map 10**

1. **Search Algorithm Description** 
   1. **Input and Output Specification**
   2. **State Representation and Visited Table**
   3. **Solvers explaination**
      1. **BFS Solver**
      2. **DFS Solver**
      3. **UCS Solver**
      4. **A\* Solver**
2. **Experiments** 
   1. **Search Time**
      1. **Measurements**
      2. **Visualizations**
      3. **Insight**
   2. **Memory Usage**
      1. **Measurements**
      2. **Visualizations**
      3. **Insight**
   3. **Expanoded** 
      1. **Measurements**
      2. **Visualizations**
      3. **Insight**
3. **Project Plan and Task Distribution** 
   1. **Project Plan**
   2. **Task Distribution**
4. **References**
5. **Apendix**