**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**NHÓM N930215**

**BÁO CÁO ĐỒ ÁN 1**

**LẬP TRÌNH GAME BẰNG NGÔN NGỮ PYTHON**

**Môn học: Thực hành Nhập môn công nghệ thông tin 2**

Thành phố Hồ Chí Minh – 2019

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**NHÓM N930215**

**BÁO CÁO ĐỒ ÁN 1**

**| Đề tài |**

**LẬP TRÌNH GAME BẰNG NGÔN NGỮ PYTHON**

**Môn học: Thực hành Nhập môn công nghệ thông tin 2**

**| Giáo viên hướng dẫn |**

**Thầy Võ Hoàng Quân**

Thành phố Hồ Chí Minh – 2019

|  |
| --- |
| **LỜI CẢM ƠN** |

Trong thời gian 1 tháng thực hiện đồ án, nhóm chúng em đã có cơ hội học hỏi và vận dụng nhiều kiến thức mới, bổ ích cũng những kinh nghiệm thực tế ngay khi còn là sinh viên năm nhất. Những kĩ năng tự học, tự tìm tòi và làm việc nhóm mà chúng em rèn luyện được qua đồ án thật sự là những kinh nghiệm quý báu đối với chúng em. Để có được những trải nghiệm ấy, chúng em cám ơn thầy Đỗ Hoàng Quân đã nhiệt tình hướng dẫn và tạo điều kiện cho chúng em có.

Trong quá trình thực hiện đồ án và làm báo cáo, chắc chắn sẽ không tránh khỏi những sai sót, mong thầy bỏ qua và góp ý để chúng em học thêm được nhiều kinh nghiệm và hoàn thành tốt hơn những đồ án sau này, nhất là những công việc sau này.

Chúng em xin chân thành cám ơn!

|  |
| --- |
| **MỤC LỤC** |

[**LỜI CẢM ƠN** 1](#_Toc6789783)

[**MỤC LỤC** 2](#_Toc6789784)

[**DANH MỤC HÌNH** 4](#_Toc6789785)

[**DANH MỤC BẢNG** 5](#_Toc6789786)

[**CHƯƠNG I: GIỚI THIỆU** 6](#_Toc6789787)

[1.1. Giới thiệu nhóm 6](#_Toc6789788)

[1.1.1. Danh sách thành viên 6](#_Toc6789789)

[1.1.2. Mục tiêu hoạt động 6](#_Toc6789790)

[1.2. Giới thiệu đồ án 1 6](#_Toc6789791)

[**CHƯƠNG II: NỘI DUNG ĐỒ ÁN** 7](#_Toc6789792)

[2.1. Các chức năng cơ bản của game 7](#_Toc6789793)

[2.1.1. Kĩ Thuật 7](#_Toc6789794)

[2.1.2. Dữ liệu 7](#_Toc6789795)

[2.1.3. Nội dung – hình thức 7](#_Toc6789796)

[2.1.4. Đồ họa 7](#_Toc6789797)

[2.1.4. Âm thanh 8](#_Toc6789798)

[2.2. Các chức năng nâng cao của game 8](#_Toc6789799)

[2.2.1. Lưu trữ bằng tài khoản 8](#_Toc6789800)

[2.2.2. Mã hóa dữ liệu 8](#_Toc6789801)

[2.2.3. Giao diện đăng nhập, đăng kí 8](#_Toc6789802)

[2.2.4. Minimap 8](#_Toc6789803)

[2.2.5. Bảng xếp hạng thời gian thực 9](#_Toc6789804)

[2.2.6. Hệ thống Camera View 9](#_Toc6789805)

[2.2.7. Cửa hàng 9](#_Toc6789806)

[2.2.8. Bùa lợi bổ sung 9](#_Toc6789807)

[2.3. Mã nguồn game 10](#_Toc6789808)

[2.3.1. Thư viện có sẵn 10](#_Toc6789809)

[2.3.2. Mã nguồn tham khảo 10](#_Toc6789810)

[2.3.3. Cấu trúc mã nguồn 11](#_Toc6789811)

[**PHỤ LỤC** 45](#_Toc6789812)

[**TÀI LIỆU THAM KHẢO** 46](#_Toc6789813)

|  |
| --- |
| **DANH MỤC HÌNH** |

Hình 2.3.1: Thư mục gốc trò chơi

|  |
| --- |
| **DANH MỤC BẢNG** |

|  |
| --- |
| **CHƯƠNG I: GIỚI THIỆU** |

## 1.1. Giới thiệu nhóm

Tên nhóm: N930215

Ngày thành lập: 13/03/2019

### 1.1.1. Danh sách thành viên

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ và tên** | **Email** | **SĐT** |
| 1 | 18120339 | Hà Văn Duy | viplazylmht@gmail.com | 0396161438 |
| 2 | 18120333 | Nguyễn Quốc Dũng | njkid555@gmail.com | 0896716226 |
| 3 | 18120320 | Bùi Hữu Đức | buihuuduc233@gmail.com | 0936902402 |
| 4 | 18120322 | Lưu Thiện Đức | ducluuthien@gmail.com | 0905804139 |
| 5 | 18120321 | Huỳnh Thanh Đức | thanhduc4320@gmail.com | 0869395072 |
| 6 | 18120345 | Phạm Minh Duy | minhduy.bthuy@gmail.com | 0342569003 |

Bảng 1.1.1

### 1.1.2. Mục tiêu hoạt động

Cùng nhau học tập, nghiên cứu, trau dồi kiến thức, kĩ năng để hoàn thành đồ án 1 và 2 bộ môn “Thực hành Nhập môn Công Nghệ Thông Tin 2”.

## 1.2. Giới thiệu đồ án 1

* Nội dung: Xây dựng một game đua xe đơn giản có đặt cược.
* Quy định : Sử dụng ngôn ngữ Python.
* Tên game: “The Amazing Race”

|  |
| --- |
| **CHƯƠNG II: NỘI DUNG ĐỒ ÁN** |

## 2.1. Các chức năng cơ bản của game

### 2.1.1. Kĩ Thuật

* Ngôn ngữ: Sử dụng ngôn ngữ Python để lập trình game.
* Thư viện: Được phép sử dụng thư viện.

### 2.1.2. Dữ liệu

* Có thông tin người chơi: Số tiền hiện có, lịch sử cược, …
* Có số tiền cược.

### 2.1.3. Nội dung – hình thức

Chủ đề: Cá cược một cuộc đua trong đó bao gồm mô phỏng nhiều đối tượng đua trên các làn đua, cụ thể:

* Người chơi đăng nhập vào trò chơi với thông tin cá nhân.
* Người chơi nhập mức tiền cược và chọn đối tượng cược là thắng trước khi cuộc đua diễn ra.
* Có tùy chọn thay đổi đường đua.
* Có tùy chọn thay đổi các bộ đối tượng đua.
* Trong quá trình đua, ngẫu nhiên tạo ra các bùa trên đường đua để thay đổi tốc độ, trạng thái của các đối tượng đua trong một khoảng thời gian tối thiểu: bùa dừng lại, bùa tăng tốc, bùa quay đầu, bùa quay về vị trí xuất phát, bùa bay thẳng về đích.
* Tiếp tục cuộc đua cho đến khi đối tượng sau cùng về đích, có bảng xếp hạng.
* Nếu cược thắng, người chơi sẽ được gấp đôi tiền cược, nếu thua thì sẽ bị mất số tiền cược.
* Người chơi được nạp thêm tiền.
* Cho phép chơi tiếp (nếu còn tiền) đến khi người chơi muốn kết thúc trò chơi.

### 2.1.4. Đồ họa

* Thay đổi kích thước game phù hợp với màn hình.
* Hình ảnh không bị bản quyền hoặc tự thiết kế.
* Có tối đa năm bộ đối tượng đua khác nhau để người chơi tùy chọn, trong đó có một bộ là hình các thành viên trong nhóm.
* Đối tượng về đích đầu tiên có động tác ăn mừng chiến thắng.
* Sau khi cuộc đua kết thúc, hiển thị bảng xếp hạng, hiển thị người chơi thắng hay thua và tình trạng tài khoản sau cuộc đua.

### 2.1.4. Âm thanh

* Nhạc nền.
* Nhạc đua.
* Nhạc kết thúc.

## 2.2. Các chức năng nâng cao của game

### 2.2.1. Lưu trữ bằng tài khoản

Cho phép người chơi đăng kí tài khoản cá nhân, dùng tài khoản đó để đăng nhập vào trò chơi. Các dữ liệu của người chơi như tình trạng tiền, lịch sử chơi, số lượng các item, … sẽ được lưu trữ vào file để hỗ trợ các lần chơi sau.

### 2.2.2. Mã hóa dữ liệu

Các thông tin tài khoản trước khi lưu vào file được mã hóa theo một cơ chế đặc biệt để đảm bảo bảo mật dữ liệu, tránh bị rò rỉ và thay đổi thông tin dễ dàng.

### 2.2.3. Giao diện đăng nhập, đăng kí

Hiển thị giao diện đăng nhập (đăng kí) với tên tài khoản và mật khẩu (đã được đổi thành dấu \* khi nhập). Với trong tin đó người chơi có thể đăng nhập tiếp vào tài khoản cũ hoặc tạo một tài khoản mới dễ dàng.

### 2.2.4. Minimap

Hiển thị quá trình đua một cách khái quát. Minimap là một bản đồ nhỏ chứa vị trí của người chơi và của sổ trên đường đua được cập nhật thời gian thực. Minimap cho phép người chơi xem vị trí hiện tại của đối tượng họ đặt cược, đồng thời cho biết đã có ai tới đích hay chưa.

### 2.2.5. Bảng xếp hạng thời gian thực

Hiển thị các thứ hạng trong trận đua một cách chi tiết nhất có thể, chạy theo thời gian thực.

### 2.2.6. Hệ thống Camera View

Cho phép người chơi điều chỉnh tầm nhìn vào một đối tượng trên một làn đua cụ thể. Điều này thuận lợi cho quá trình theo dõi chi tiết cuộc đua với khoảng cách lớn.

### 2.2.7. Cửa hàng

Là nơi để người chơi nạp thêm tiền vào tài khoản và mua bùa lợi.

### 2.2.8. Bùa lợi bổ sung

* Bùa giảm tốc độ: hạn chế tốc độ trong một khoảng thời gian.
* Bùa tốc biến: dịch chuyển một đoạn ngắn lên phía trước.
  + Bùa giải/khiên: Kích hoạt giúp bỏ qua một bùa tiếp theo trên đường đua.
  + Bùa ngôi sao hi vọng: Dùng để tăng mức tiền thưởng, hữu dụng khi người chơi đặt cược lớn hoặc khi gần hết tiền.

Người chơi quyết định có sử dụng hai loại bùa giải và bùa ngôi sao hi vọng hay không trước khi trận đua bắt đầu.

## 2.3. Mã nguồn game

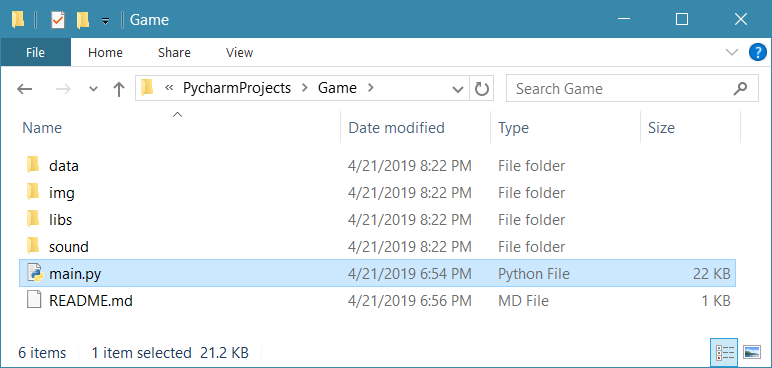
### 2.3.1. Thư viện có sẵn

1. Pygame: Thư viện dùng để đồ họa giao diện, tương tác với thiết bị nhập, xuất.
2. Colormap: sử dụng hàm hex2rgb: chuyển đổi mã màu từ hệ Hexa (kiểu String) RGB code (kiểu list).
3. Time: Thư viện dùng để lấy thời gian hiện tại của đồng hồ hệ thống.
4. Random: Tạo các số ngẫu nhiên .
5. Os: Thao tác với các câu lệnh, môi trường hệ điều hành.
6. Math: Thư viện toán học, dùng để tính toán .

### 2.3.2. Mã nguồn tham khảo

### 2.3.3. Cấu trúc mã nguồn

Toàn cảnh cấu trúc của mã nguồn của trò chơi:



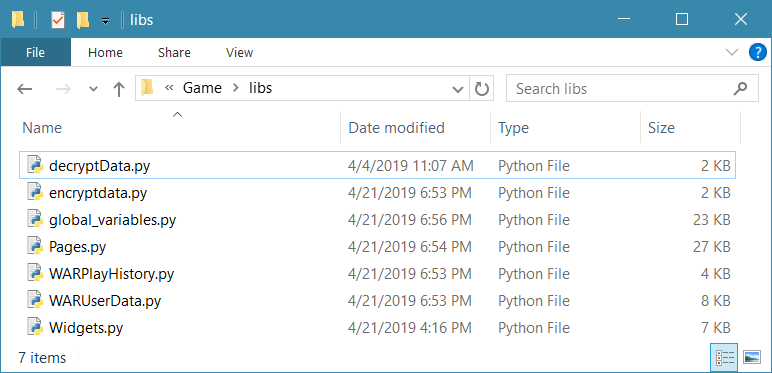
Hình 2.3.1

* data: Thư mục chứa dữ liệu trò chơi, bao gồm lưu cài đặt, thông tin người chơi, lịch sử chơi.
* img: Thư mục chứa toàn bộ hình ảnh phục vụ cho trò chơi.
* libs: Thư mục chứa các file mã nguồn con cho trò chơi.
* sound: Thư mục chứa các tập tin âm thanh.
* main.py: File dùng để khởi chạy chương trình chính.
* README.md: File chứa thông tin về trò chơi.

File main.py: Chương trình chính

001 **from** libs.global\_variables **import** \*  
002 **from** libs.Widgets **import** \*  
003 **from** libs.Pages **import** \*  
004 **from** libs.WARUserData **import** \*  
005 **from** libs.WARPlayHistory **import** \*  
006 **import** pygame  
007 **import** time  
008   
009 **def** loginActivity():  
010 listUser = []  
011 user = User()  
012 readSuccess = ReadUsersData.GetAllUsersData(listUser)  
013 gameLancher.draw\_map(2)  
014 loginPage = LoginPage(gameLancher)  
015 loginPage.drawLoginPage()  
016 btn\_signin = loginPage.btn\_signin  
017 btn\_signup = loginPage.btn\_signup  
018 warning = loginPage.warningText  
019 pygame.display.flip()  
020 loginPage.passWordInput.isPassword = **True**021 input\_boxes = [loginPage.userNameInput, loginPage.passWordInput]  
022 **if** (readSuccess == -2):  
023 warning.setText(**"You have edit UsersData file so all your data is deleted"**)  
024 listUser = []  
025 **while True**:  
026 gameLancher.draw\_map(100) **and** pygame.display.flip()  
027 **if** pygame.key.get\_pressed()[pygame.K\_F12]:  
028 gameLancher.SCREEN = pygame.display.set\_mode((gameLancher.GAME\_WIDTH,  
029 gameLancher.GAME\_HEIGHT), pygame.FULLSCREEN)  
030 **if** pygame.key.get\_pressed()[pygame.K\_F11]:  
031 gameLancher.SCREEN = pygame.display.set\_mode((gameLancher.GAME\_WIDTH,  
 gameLancher.GAME\_HEIGHT))  
032 **if** btn\_signin.is\_clicked(gameLancher):  
033 showPass = loginPage.passWordInput.text  
034 user.name = loginPage.userNameInput.text  
035 **if** loginPage.passWordInput.isPassword:  
036 user.password = loginPage.passWordInput.hidetext  
037 **else**:  
038 user.password = loginPage.passWordInput.text  
039 **if** (len(user.name) == 0) | (len(user.password) == 0):  
040 warning.setText(**"Please enter your username and password"**)  
041 **elif** len(user.password) != len(showPass):  
042 warning.setText(**"Please do not user your numpad to enter password"**)  
043 **else**:  
044 exitCode = LoginCore.FindUserName(listUser, user)  
045 **if** ((exitCode == -1)):  
046 warning.setText(**"Account is not signed up yet."**)  
047 **elif** ((exitCode == -2)):  
048 warning.setText(**"Password is wrong."**)  
049 **else**:  
050 welcome = **"Welcome back "** + str(user.name)  
051 warning.setText(welcome)  
052 time.sleep(0.08)  
053 gameLancher.IS\_SIGNED\_IN = **True**054 loginPage.drawLoginPage()  
055 pygame.display.flip()  
056 **return** exitCode, listUser  
057 **if** btn\_signup.is\_clicked(gameLancher):  
058 **if** loginPage.passWordInput.isPassword:  
059 user.password = loginPage.passWordInput.hidetext  
060 **else**:  
061 user.password = loginPage.passWordInput.text  
062 showPass = loginPage.passWordInput.text  
063 user.name = loginPage.userNameInput.text  
064 i = 0  
065 isVN = **False**066 **while** i < len(user.password):  
067 **if** user.password[i] > **'~' or** user.password[i] == **''**:  
068 isVN = **True**069 i += 1  
070 i = 0  
071 **while** i < len(user.name):  
072 **if** user.name[i] > **'~' or** user.name[i] == **''**:  
073 isVN = **True**074 i += 1  
075 **if** isVN:  
076 warning.setText(**"Please turn off your unikey"**)  
077 **elif not** LoginCore.CheckName(user):  
078 warning.setText(**"Your username and password cannot have \" symbol"**)  
079 **elif** len(user.password) != len(showPass):  
080 warning.setText(**"Please do not user your numpad to enter password"**)  
081 **elif** len(user.password) < 4:  
082 warning.setText(**"Your password must have at least 4 character"**)  
083 **elif** (len(user.name) == 0) | (len(user.password) == 0):  
084 warning.setText(**"Please enter your username and password"**)  
085 *# if account is not exist yet*086 **elif** ((LoginCore.FindUserName(listUser, user) == -1)):  
087 welcome = **"Welcome "** + str(user.name)  
088 warning.setText(welcome)  
089 user.ID = int(gameLancher.DEFAULT\_ACCOUNT\_POS)  
090 gameLancher.DEFAULT\_ACCOUNT\_POS = str(user.ID + 1)  
091 gameLancher.update\_setting\_pref()  
092 *# ADD DEFAULT DATA OF NEW USER*093 user.coins = **"100000"**094 user.playTime = **"0"**095 user.winrate = **"0"**096 listUser.append(user)  
097 WriteUsersData.WriteAllUsersData(listUser)  
098 loginPage.drawLoginPage()  
099 pygame.display.flip()  
100 **return** len(listUser) - 1, listUser  
101 **else**:  
102 warning.setText(**"Account is already exist."**)  
103   
104 **for** event **in** pygame.event.get():  
105 **if** event.type == pygame.QUIT:  
106 pygame.quit()  
107 exit(0)  
108 loginPage.loginHandle(event)  
109 loginPage.drawLoginPage()  
110   
111 **for** box **in** input\_boxes:  
112 box.draw(gameLancher.SCREEN)  
113 pygame.display.flip()  
114   
115 **def** LoadUserHistory(user, history):  
116 ReadHistoryData.GetAllHistoryData(user.ID, history)  
117   
118 **def** main\_game(listUser, userIndex, history):  
119 user = listUser[userIndex]  
120 timenow = int(round(time.time()\*1000))  
121 lasttime = timenow  
122 gameLancher.START\_POS = gameLancher.GAME\_WIDTH/3  
123 racers = gameLancher.assign\_racers()  
124 ranking = Ranking(gameLancher, racers)  
125 minimap = Minimap(gameLancher)  
126 camera = Camera(gameLancher)  
127 mainpage = MainPage(gameLancher)  
128 settingPage = SettingPage(gameLancher)  
129 historyPage = HistoryPage(gameLancher)  
130 shoppage = Shoppage(gameLancher, user)  
131   
132 finish = **False**133 play = **False**134 rollback = 100  
135 gameLancher.draw\_map(100) **and** pygame.display.flip()  
136 time\_count = 2001  
137 coin\_input = 0  
138 COUNT\_AMULET = 0  
139 sound\_result = **False**140 time\_amulet\_appear = 3  
141 count = 6  
142 time\_amulet = 0  
143 racer\_play\_pos = 0  
144   
145 use\_star = **False**146 use\_shield = **False**147 used\_shield = **False**148   
149 **while not** finish:  
150 **if** pygame.key.get\_pressed()[pygame.K\_F12]:  
151 gameLancher.SCREEN = pygame.display.set\_mode((gameLancher.GAME\_WIDTH,  
152 gameLancher.GAME\_HEIGHT), pygame.FULLSCREEN)  
153 **if** pygame.key.get\_pressed()[pygame.K\_F11]:  
154 gameLancher.SCREEN = pygame.display.set\_mode((gameLancher.GAME\_WIDTH,  
 gameLancher.GAME\_HEIGHT))  
155 *###########################*156 timenow = int(round(time.time()\*1000))  
157 **if** timenow - lasttime > gameLancher.TIME\_INTERVAL:  
158 *# Xu ly giao dien sau moi Frame (FPS)*159 time\_count += gameLancher.TIME\_INTERVAL  
160 **if** time\_count > 1000:  
161 *# Xu ly su kien sau moi 1 giay (thay doi toc do racer)*162 time\_count -= 1000  
163 **for** r **in** racers:  
164 r.updatespeed()  
165 **if** play:  
166 rollback += camera.delta  
167 gameLancher.draw\_map(rollback)  
168 *####END CHECK TIME####*169 **if** gameLancher.IS\_IN\_SHOP:  
170 shoppage.DrawShop(user)  
171 **if** shoppage.price\_shield.is\_clicked(gameLancher) **and** \  
 int(user.coins) >= shoppage.val\_price\_shield:  
172 user.coins = str(int(user.coins) - shoppage.val\_price\_shield)  
173 user.item\_shield += 1  
174 listUser[userIndex] = user  
175 WriteUsersData.WriteAllUsersData(listUser)  
176 time.sleep(0.2)  
177   
178 **if** shoppage.buy\_coin.is\_clicked(gameLancher):  
179 user.coins = str(int(user.coins) + 10000)  
180 listUser[userIndex] = user  
181 WriteUsersData.WriteAllUsersData(listUser)  
182 time.sleep(0.2)  
183   
184 **if** shoppage.btn\_back.is\_clicked(gameLancher):  
185 gameLancher.IS\_IN\_SHOP = **False**186 *####END SHOP CHECK####*187 infoZone = InfoZone(gameLancher, listUser[userIndex])  
188 infoZone.drawInfoZoneExpand() **if** infoZone.is\_clicked() **else** infoZone.drawInfoZone()  
189 *####END INFO ZONE DRAW####*190 **if** play **and** use\_shield:  
191 gameLancher.IC\_SHIELD.draw(gameLancher.SCREEN)  
192 **if** gameLancher.IC\_SHIELD.is\_clicked() **and not** used\_shield:  
193 used\_shield = **True**194 print(**"SHIELD CLICKED!"**)  
195   
196 **if** use\_shield **and** used\_shield:  
197 racers[racer\_play\_pos].exist\_shield\_amulet = **True**198 use\_shield = **False**199   
200 **if** play **and** use\_star:  
201 racers[racer\_play\_pos].exist\_star\_amulet = **True**202 use\_star = **False**203 user.use\_star = **True**204 *####END CHECK CUSTOM AMULET####*205 **if** gameLancher.IS\_IN\_SETTINGS:  
206 settingPage.drawSettingPage()  
207 **if** settingPage.btn\_setplayer.is\_clicked(gameLancher):  
208 time.sleep(0.1)  
209 gameLancher.DEFAULT\_RACERS\_CODE = str(settingPage.drawChooseRacer())  
210 gameLancher.update\_setting\_pref()  
211 racers = gameLancher.assign\_racers()  
212 gameLancher.IS\_IN\_SETTINGS = **True**213 **if** settingPage.btn\_setmap.is\_clicked(gameLancher):  
214 time.sleep(0.1)  
215 gameLancher.DEFAULT\_MAP\_CODE = str(settingPage.drawChooseMap())  
216 gameLancher.update\_setting\_pref()  
217 gameLancher.assign\_map()  
218 gameLancher.IS\_IN\_SETTINGS = **True**219 **if** settingPage.btn\_modsound.is\_clicked(gameLancher):  
220 time.sleep(0.1)  
221 gameLancher.DEFAULT\_SOUND\_CODE = settingPage.drawOptionSound()  
222 **if** gameLancher.DEFAULT\_SOUND\_CODE[0]:  
223 pygame.mixer.music.load(**"sound/theme\_song\_cut.mp3"**)  
224 pygame.mixer.music.play(-1)  
225 **else**:  
226 pygame.mixer.music.stop()  
227 gameLancher.update\_setting\_pref()  
228 gameLancher.IS\_IN\_SETTINGS = **True**229 **if** settingPage.btn\_back.is\_clicked(gameLancher):  
230 gameLancher.IS\_IN\_SETTINGS = **False**231 time.sleep(0.1)  
232 *####END CHECK SUB SETTING####*233 **if not** gameLancher.IS\_GAME\_PLAYING **and not** gameLancher.IS\_IN\_SETTINGS \  
 **and not** gameLancher.IS\_IN\_HISTORY **and not** gameLancher.IS\_IN\_SHOP:  
234 mainpage.drawMainPage()  
235 **if** mainpage.btn\_start.is\_clicked(gameLancher) **and not** play:  
236 time.sleep(0.1)  
237 isOK, coin\_input, distance, racer\_play\_pos, use\_star, \  
 use\_shield = mainpage.drawInitStart(user, racers)  
238 camera.follow = racer\_play\_pos  
239 **if** isOK:  
240 gameLancher.IS\_GAME\_PLAYING = **True**241 play = **True**242 pygame.mixer.music.load(**"sound/fast\_lane.mp3"**)  
243 **if** gameLancher.DEFAULT\_SOUND\_CODE[1]:  
244 pygame.mixer.music.play(-1)  
245 gameLancher.DISTANCE = distance\*gameLancher.SCALE\_X  
246 gameLancher.DISTANCE\_DEFAULT = distance\*gameLancher.SCALE\_X  
247 **if** use\_shield:  
248 user.item\_shield -= 1  
249   
250 **if** mainpage.btn\_logout.is\_clicked(gameLancher) **and not** play:  
251 listUser[userIndex] = user  
252 WriteUsersData.WriteAllUsersData(listUser)  
253 gameLancher.IS\_SIGNED\_IN = **False**254 time.sleep(0.1)  
255 destroyClass(mainpage, racers, ranking, minimap, \  
 camera, settingPage, historyPage, shoppage)  
256 **return** main()  
257   
258 **if** mainpage.btn\_store.is\_clicked(gameLancher) **and not** play:  
259 gameLancher.IS\_IN\_SHOP = **True**260 play = **False**261 **if** mainpage.btn\_setting.is\_clicked(gameLancher) **and not** play:  
262 gameLancher.IS\_IN\_SETTINGS = **True**263 time.sleep(0.1)  
264 play = **False**265   
266 **if** mainpage.btn\_history.is\_clicked(gameLancher) **and not** play:  
267 gameLancher.IS\_IN\_HISTORY = **True**268 historyPage.setHistory(history)  
269 play = **False**270 *####END CHECK MAINPAGE####*271 **if** gameLancher.IS\_IN\_HISTORY:  
272 **if** (len(history) != 0):  
273 **if** pygame.key.get\_pressed()[pygame.K\_UP]:  
274 historyPage.Up()  
275 time.sleep(0.1)  
276 historyPage.setHistory(history)  
277 **if** pygame.key.get\_pressed()[pygame.K\_DOWN]:  
278 historyPage.Down()  
279 time.sleep(0.1)  
280 historyPage.setHistory(history)  
281 **if** historyPage.btn\_back.is\_clicked(gameLancher):  
282 gameLancher.IS\_IN\_HISTORY = **False**283 time.sleep(0.1)  
284 historyPage.draw(history)  
285 *####END CHECK HISTORY####*286 isScrolling = **False**287 last\_racer = **False**288 winner = racers[0]  
289 **for** r **in** racers:  
290 **if** play:  
291 **if** (time.clock() > time\_amulet\_appear **and** COUNT\_AMULET < count):  
292 **if not** r.exist\_amulet:  
293 r.Amulet\_appear()  
294 COUNT\_AMULET += 1  
295 **if** (COUNT\_AMULET == count):  
296 time\_amulet\_appear += 8  
297 count += 6  
298 **if** (r.exist\_amulet):  
299 **if** (r.x > r.amulet\_x **and** r.time > 0):  
300 r.active()  
301 time\_amulet = time.clock() + 2  
302 **elif** (r.time > 0):  
303 **if** (time.clock() < time\_amulet):  
304 r.active()  
305 **else**:  
306 r.kind = 0  
307 r.exist\_turn = **False**308 **else**:  
309 r.kind = 0  
310 r.exist\_turn = **False**311 r.exist\_star\_amulet = **False**312 **if** r.rank == 1:  
313 winner = r  
314 **if** r.rank == 6 **and** r.x >= gameLancher.DISTANCE:  
315 last\_racer = **True**316 r.draw\_amulet(camera.delta)  
317 *# Kiem tra ket thuc cuoc dua*318 isScrolling = r.update(camera) **or** isScrolling  
319 *# Check to show shield*320 use\_shield = use\_shield **and** racers[racer\_play\_pos].update(camera)  
321 *####END DRAW RACER IN GAME PLAY####*322 **if** gameLancher.IS\_GAME\_PLAYING:  
323 ranking.update(racers)  
324 minimap.update(racers, racer\_play\_pos, camera.follow)  
325 camera.update(racers[camera.follow])  
326 **if not** isScrolling:  
327 coinResult,finish\_r,temp\_user,want\_play = \  
 finish\_race(gameLancher,winner,racers[racer\_play\_pos],user,coin\_input)  
328 **if** finish\_r:  
329 gameLancher.IS\_GAME\_PLAYING = isScrolling  
330 play = isScrolling  
331 win = 0  
332 lose = 0  
333 **for** i **in** history:  
334 **if** (i.coinResult[0] == **'+'**):  
335 win += 1  
336 **else**:  
337 lose += 1  
338 **if** (coinResult[0] == **'+'**):  
339 win += 1  
340 **else**:  
341 lose += 1  
342 temp\_user.winrate = str(int((win/(win + lose))\*100))  
343 user = temp\_user  
344 listUser[userIndex] = user  
345 WriteUsersData.WriteAllUsersData(listUser)  
346 currentPlay = History()  
347 currentPlay.racerType = str(gameLancher.DEFAULT\_RACERS\_CODE) \  
 + str(racer\_play\_pos)  
348 currentPlay.coinResult = coinResult  
349 history.append(currentPlay)  
350 WriteHistoryData.WriteAllHistoryData(user.ID, history)  
351 destroyClass(mainpage, racers, ranking, minimap, \  
 camera, settingPage, historyPage, shoppage)  
352 **if** want\_play:  
353 **return** main\_game(listUser, userIndex, history)  
354 **else**:  
355 gameLancher.IS\_SIGNED\_IN = **False**356 **return** main()  
357 *####END DRAW WHILE GAME PLAY####*358 **if** last\_racer:  
359 ranking.show\_top1 = **True**360 **if** ranking.y < gameLancher.GAME\_HEIGHT/3.5:  
361 ranking.y += 3  
362 **if not** sound\_result:  
363 sound\_result = **True**364 *# music*365 pygame.mixer.music.load(**"sound/all\_stop\_now.mp3"**)  
366 **if** gameLancher.DEFAULT\_SOUND\_CODE[0]:  
367 pygame.mixer.music.play(-1)  
368 *# /music*369 **if** gameLancher.DEFAULT\_SOUND\_CODE[2]:  
370 **if** winner.num == racer\_play\_pos:  
371 sound = pygame.mixer.Sound(**'sound/win.wav'**)  
372 sound.play()  
373 **else**:  
374 sound = pygame.mixer.Sound(**'sound/lose.wav'**)  
375 sound.play()  
376 **if** play:  
377 gameLancher.DISTANCE += camera.delta  
378 **for** event **in** pygame.event.get():  
379 **if** event.type == pygame.QUIT:  
380 finish = **True**381 pygame.display.flip()  
382 gameLancher.clock.tick(gameLancher.FPS)  
383 *####END CHECK GAME ENDED####*384 **def** main():  
385 *# music*386 **if** gameLancher.DEFAULT\_SOUND\_CODE[0]:  
387 pygame.mixer.music.load(**"sound/theme\_song\_cut.mp3"**)  
388 pygame.mixer.music.play(-1)  
389 *# /music*390 user = User()  
391 listUser = []  
392 userIndex = 0  
393 **if not** gameLancher.IS\_SIGNED\_IN:  
394 userIndex, listUser = loginActivity()  
395 user = listUser[userIndex]  
396 time.sleep(0.5)  
397 **else**:  
398 listUser.append(user)  
399 history = []  
400 LoadUserHistory(user, history)  
401 main\_game(listUser, userIndex, history)  
402   
403 **def** finish\_race(game, racer, player\_choose, user, coin\_input):  
404 game.SCREEN.blit(game.IC\_RESULT\_BOARD, resultRect)  
405 game.SCREEN.blit(player\_choose.imgbig, (resultRect.x + \  
 resultRect.w\*0.3,resultRect.y + resultRect.h\*0.3))  
406 last\_coin = int(user.coins)  
407 change\_coin = int(coin\_input)  
408 current\_coin = int(user.coins)  
409   
410 **if** racer.num == player\_choose.num:  
411 game.SCREEN.blit(game.IC\_WIN, (resultRect.x + resultRect.w//2,  
 resultRect.y + resultRect.h\*0.2))  
412 *# set win state include VAT 5%*413 **if** user.use\_star:  
414 change\_coin = change\_coin\*2  
415 current\_coin = str(last\_coin + change\_coin - change\_coin//20)  
416 last\_coin = str(last\_coin)  
417 change\_coin = **"+"** + str(change\_coin - change\_coin//20)  
418 **else**:  
419 game.SCREEN.blit(game.IC\_LOSE, (resultRect.x + resultRect.w//2,  
 resultRect.y + resultRect.h\*0.2))  
420 *# set lose state*421 **if** user.use\_star:  
422 change\_coin = int(change\_coin\*1.5)  
423 current\_coin = str(max(last\_coin - change\_coin, 0))  
424 change\_coin = **"-"** + str(last\_coin - int(current\_coin))  
425 last\_coin = str(last\_coin)  
426   
427 game.list\_tv[0].setText(**"LAST COIN: "** + last\_coin)  
428 game.list\_tv[1].setText(**"CHANGE: "** + change\_coin)  
429 game.list\_tv[2].setText(**"CURRENT COINS: "** + current\_coin)  
430 **for** tv **in** game.list\_tv:  
431 tv.show()  
432 gameLancher.btn\_end.show()  
433 gameLancher.btn\_play\_again.show()  
434 **if** gameLancher.btn\_end.is\_clicked(gameLancher):  
435 **if** gameLancher.DEFAULT\_SOUND\_CODE[0]:  
436 pygame.mixer.music.load(**"sound/theme\_song\_cut.mp3"**)  
437 pygame.mixer.music.play(-1)  
438 new\_user = User()  
439 user.cloneTo(temp\_user=new\_user)  
440 new\_user.coins = current\_coin  
441 new\_user.playTime = str(int(new\_user.playTime) + 1)  
442 **return** change\_coin, **True**, new\_user, **False**443   
444 **if** gameLancher.btn\_play\_again.is\_clicked(gameLancher):  
445 **if** gameLancher.DEFAULT\_SOUND\_CODE[0]:  
446 pygame.mixer.music.load(**"sound/theme\_song\_cut.mp3"**)  
447 pygame.mixer.music.play(-1)  
448 gameLancher.IS\_GAME\_PLAYING = **False**449 gameLancher.IS\_IN\_SETTINGS = **False**450 gameLancher.IS\_IN\_HISTORY = **False**451 new\_user = User()  
452 user.cloneTo(temp\_user=new\_user)  
453 new\_user.coins = current\_coin  
454 new\_user.playTime = str(int(new\_user.playTime)+1)  
455 **return** change\_coin, **True**, new\_user, **True**456   
457 **return** change\_coin, **False**, **None**, **False**458 *####INIT GAME####*459 **def** destroyClass(\*args):  
460 **for** i **in** args:  
461 **del** i  
462 gameLancher = INIT\_GAME()  
463 resultRect = gameLancher.IC\_RESULT\_BOARD.get\_rect()  
464 resultRect.center = (gameLancher.GAME\_WIDTH//2, gameLancher.GAME\_HEIGHT//2)  
465   
466 main()  
467   
468 pygame.quit()  
469 **del** gameLancher  
470 exit()

Các file mã nguồn con:



Hình 2.3.2

*File decryptData.py:*Giải mã dữ liệu

01 *# giai ma du lieu*   
02 numBitsRanded = 3  
03 **class** decrypt:  
04 **def** Convert(a, strBit):  
05 bitA = ord(a)  
06 *#add to string*07 i = 0  
08 tempt = bitA  
09   
10 *#get rid randedBit*11 temp = bitA >> (8 - numBitsRanded)  
12 temp = temp << (8 - numBitsRanded)  
13 bitA ^= temp  
14   
15 **while**(i < (8 - numBitsRanded)):  
16 temp = bitA >> (8 - i - numBitsRanded - 1)  
17 a = str(temp)  
18 strBit += a  
19 temp = temp << (8 - i - numBitsRanded - 1)  
20 bitA ^= temp  
21 i += 1  
22 **return** strBit  
23   
24 **def** XorData(currentBits, key):  
25 i = 0  
26 bits = 0  
27 **while**(i < len(currentBits)):  
28 bits = bits << 1  
29 bits += int(currentBits[i])  
30 i += 1  
31 *#xored key*32 bits ^= key  
33 a = str(chr(bits))  
34 **return** a  
35   
36 **def** DecyptData(data, key):  
37 strBit = **""**38 i = 0  
39 **while**(i < len(data)):  
40 strBit = decrypt.Convert(data[i], strBit)  
41 i += 1  
42 decryptedData = **""**43 currentBits = []  
44 i = 0  
45 **while**(i < len(strBit)):  
46 **if**((i != 0) & (i % 8 == 0)):  
47 decryptedData += decrypt.XorData(currentBits, key)  
48 currentBits = []  
49 currentBits.append(int(strBit[i]))  
50 i += 1  
51 decryptedData += decrypt.XorData(currentBits, key)  
52 **return** decryptedData  
53 *#DEMO DecyptData("G$7#DVZ[", 123)*54 **pass**

55

*File encryptdata.py:* Mã hóa dữ liệu

01 *# ma hoa du lieu*  
02 **import** random  
03 numBitsRanded = 3  
04 **class** encypt:  
05 **def** FindNeedLength(a, b):  
06 i = 1  
07 **while**(((a\*i) % b) != 0):  
08 i += 1  
09 **return** i  
10   
11 **def** ConvertAndXor(a, key, xoredData):  
12 bitA = ord(a)  
13 bitA ^= key  
14 i = 0  
15 **while**(i < 8):  
16 temp = bitA >> (8 - i - 1)  
17 a = str(temp)  
18 xoredData += a  
19 temp = temp << (8 - i - 1)  
20 bitA ^= temp  
21 i += 1  
22 **return** xoredData  
23   
24 **def** RandBits(bits):  
25 bitRand = random.randrange(1, 4)  
26 i = 0  
27 **while**(i < len(bits)):  
28 bitRand = bitRand << 1  
29 bitRand += int(bits[i])  
30 i += 1  
31 *#convert to char*32 a = str(chr(bitRand))  
33 **return** a  
34   
35 **def** EncryptData(data, key):  
36 lengthNeed = encypt.FindNeedLength(8, 8 - numBitsRanded) *#8 is bits per byte*37 **while**(len(data) % lengthNeed != 0):  
38 data += **' '**39 xoredData = **""**40 *#convert data to bit and xor with key*41 i = 0  
42 **while**(i < len(data)):  
43 xoredData = encypt.ConvertAndXor(data[i], key, xoredData)  
44 i += 1  
45 encryptedData = **""**46 currentBits = []  
47 *#encrypt*48 i = 0  
49 **while**(i < len(xoredData)):  
50 **if**((i % (8 - numBitsRanded) == 0) & (i != 0)):  
51 encryptedData += encypt.RandBits(currentBits)  
52 currentBits = []  
53 currentBits.append(int(xoredData[i]))  
54 i += 1  
55 encryptedData += encypt.RandBits(currentBits)  
56 **return** encryptedData  
57 **pass**

*File global\_variables.py*: môi trường và các biến cục bộ

001 **import** pygame  
002 **import** math  
003 **import** time  
004 **import** random  
005 **from** libs.Widgets **import** \*  
006  
007 **class** INIT\_GAME():  
008 DELAY\_TIME = 60  
009 FPS = 60  
010 clock = pygame.time.Clock()  
011 **def** \_\_init\_\_(self) -> **None**:  
012 pygame.init()  
013 pygame.display.set\_caption(**"Racing with me!"**)  
014 self.VERSION\_INFO = **"3.0 Release"**015 self.INFOR\_DISPLAY = pygame.display.Info()  
016 self.SCREEN\_SIZE = (self.INFOR\_DISPLAY.current\_w, self.INFOR\_DISPLAY.current\_h)  
017 self.GAME\_WIDTH = int(self.SCREEN\_SIZE[1]//10\*9)  
018 self.GAME\_HEIGHT = int(self.GAME\_WIDTH/3\*2)  
019 *#self.GAME\_WIDTH = 1080*020 *#self.GAME\_HEIGHT = 720*021 self.SCREEN = pygame.display.set\_mode((self.GAME\_WIDTH, self.GAME\_HEIGHT))  
022 self.GAME\_WIDTH\_DEFAULT = 1080  
023 self.GAME\_HEIGHT\_DEFAULT = 720  
024 self.SCALE\_X = self.GAME\_WIDTH/self.GAME\_WIDTH\_DEFAULT  
025 self.SCALE\_Y = self.GAME\_HEIGHT/self.GAME\_HEIGHT\_DEFAULT  
026 self.IC\_LOGOTEAM = self.load\_img(**"img/ic\_logoTeam.png"**, 70, -1)  
027 pygame.display.set\_icon(self.IC\_LOGOTEAM)  
028 self.BG\_MAP = **None**029 self.btn\_play\_again = View(self.GAME\_WIDTH/3 - self.GAME\_WIDTH/20,  
030 self.GAME\_HEIGHT/4\*3 + self.GAME\_HEIGHT/20, **"PLAY AGAIN"**, color=**"#eafd00"**)  
031 self.btn\_end = View(self.GAME\_WIDTH/3\*2 - self.GAME\_WIDTH/20,  
032 self.GAME\_HEIGHT/4\*3 + self.GAME\_HEIGHT/20, **"QUIT"**, color=**"#eafd00"**)  
033 self.list\_tv = (View(self.GAME\_WIDTH/4 - self.GAME\_WIDTH/20, self.GAME\_HEIGHT/2 + \  
 self.GAME\_HEIGHT/20\*1, color=**"#65d06f"**, gravity=**"center\_horizontal"**),  
034 View(self.GAME\_WIDTH/4 - self.GAME\_WIDTH/20, self.GAME\_HEIGHT/2 + \  
 self.GAME\_HEIGHT/20\*2, color=**"#65d06f"**, gravity=**"center\_horizontal"**),  
035 View(self.GAME\_WIDTH/4 - self.GAME\_WIDTH/20, self.GAME\_HEIGHT/2 + \  
 self.GAME\_HEIGHT/20\*3, color=**"#65d06f"**, gravity=**"center\_horizontal"**))  
036 self.IC\_RESULT\_BOARD = self.load\_img(**"img/ic\_result\_board.png"**, 800, 600)  
037 self.IC\_WIN = self.load\_img(**"img/ic\_win.png"**, 300, 300)  
038 self.IC\_LOSE = self.load\_img(**"img/ic\_lose.png"**, 300, 300)  
039 self.IC\_TOP1 = self.load\_img(**"img./ic\_top1.png"**, 0.8, 0.8)  
040 self.IC\_APP\_NAME = self.load\_img(**"img/ic\_app\_name.png"**, self.GAME\_WIDTH//2, -1)  
041 self.IC\_RANK = self.load\_img(**"img/ic\_rank.png"**,0.8,0.8)  
042 self.IC\_MINIMAP = self.load\_img(**"img/minimap.png"**, 300, 50)  
043 self.IC\_POINT\_B = self.load\_img(**"img/point\_blue.png"**, 11, 11)  
044 self.IC\_POINT\_R = self.load\_img(**"img/point\_red.png"**, 11, 11)  
045 self.IC\_MINIMAP\_CAMERA = self.load\_img(**"img/minimap\_camera.png"**, 38, 21)  
046  
047 self.IC\_STAR = ImageView(self, self.GAME\_WIDTH - self.GAME\_WIDTH//4 + 100,  
 self.GAME\_HEIGHT -120, 80, 80, **"img/ic\_star.png"**)  
048 self.IC\_SHIELD = ImageView(self, self.GAME\_WIDTH - self.GAME\_WIDTH//4,  
 self.GAME\_HEIGHT -120, 80, 80, **"img/ic\_shield.png"**)  
049 self.IC\_ARROW = self.load\_img(**"img/arrow.png"**, 50, -1)  
050 self.IC\_TICK = self.load\_img(**"img/ic\_tick.png"**, 50, -1)  
051 self.IC\_PROFILE = self.load\_img(**"img/ic\_profile1.png"**, 50, -1)  
052 self.IC\_SHIELD\_MINI = self.load\_img(**"img/ic\_shield.png"**, 35, -1)  
053 self.IC\_WINRATE = self.load\_img(**"img/ic\_win\_rate.png"**, 45, -1)  
054 self.IC\_PLAYED\_TIME = self.load\_img(**"img/ic\_playedTime.png"**, 45, -1)  
055  
056 self.IC\_COIN = self.load\_img(**"img/ic\_coin.png"**, 50, -1)  
057 self.IC\_CAMERA = (self.load\_img(**"img/camera0.png"**, 0.8, 0.8),  
058 self.load\_img(**"img/camera1.png"**, 0.8, 0.8),  
059 self.load\_img(**"img/camera2.png"**, 0.8, 0.8))  
060 self.IC\_FINISH\_FLAG = self.load\_img(**"img/line.png"**, 20, 355)  
061  
062 self.SETTING\_PREF = **"data/setting\_preferences"**063 self.ROLLBACK\_STEP = 0  
064 self.BTN\_VERSION = View(self.GAME\_WIDTH - 10, self.GAME\_HEIGHT - 10,  
 **"Versions: "**+self.VERSION\_INFO,color=**"#000000"**, gravity=**"bottom\_right"**)  
065 self.TIME\_INTERVAL = 1000/self.FPS  
066 *# some boolean*067 self.IS\_SIGNED\_IN = **False**068 self.IS\_GAME\_PLAYING = **False**069 self.IS\_GAME\_ENDED = **False**070 self.IS\_START\_OPTIONS = **False**071 self.IS\_IN\_HISTORY = **False**072 self.IS\_IN\_SETTINGS = **False**073 self.IS\_IN\_SHOP = **False**074 *# Dieu chinh quang duong dua*075 self.RESTART = **False**076 self.START\_POS = self.GAME\_WIDTH/3  
077 self.DISTANCE\_DEFAULT = 3000  
078 self.DISTANCE = self.DISTANCE\_DEFAULT  
079 *# setting file*080 self.DEFAULT\_MAP\_CODE = 1  
081 self.DEFAULT\_RACERS\_CODE = 1  
082 self.DEFAULT\_ACCOUNT\_POS = 100000  
083 *# List of sound: backgr, playing, result: 0 is off, 1 is on*084 self.DEFAULT\_SOUND\_CODE = (1, 1, 1)  
085 self.load\_setting\_pref()  
086 super().\_\_init\_\_()  
087  
088 **def** load\_img(self, link, scale\_x, scale\_y):  
089 img = pygame.image.load(link).convert\_alpha()  
090 size = img.get\_rect().size  
091 **if** scale\_x == -1:  
092 scale\_x = size[0]\*scale\_y/size[1]  
093 **if** scale\_y == -1:  
094 scale\_y = size[1]\*scale\_x/size[0]  
095 **if** scale\_x <= 3 **and** scale\_y <= 3:  
096 scale\_x = size[0]\*scale\_x  
097 scale\_y = size[1]\*scale\_y  
098 img = pygame.transform.scale(img,  
 (int(scale\_x\*self.SCALE\_X), int(scale\_y\*self.SCALE\_Y)))  
099 **return** img  
100  
101 **def** draw\_map(self, rollback):  
102 self.START\_POS = self.GAME\_WIDTH/3 + rollback  
103 **for** i **in** range(0, 2):  
104 self.SCREEN.blit(self.BG\_MAP,  
 (self.GAME\_WIDTH\*((-rollback//self.GAME\_WIDTH)+i) + rollback, 0))  
105 self.SCREEN.blit(self.IC\_FINISH\_FLAG, (self.START\_POS, 255\*self.SCALE\_Y))  
106 self.SCREEN.blit(self.IC\_FINISH\_FLAG,  
 (self.DISTANCE-50\*self.SCALE\_X, 255\*self.SCALE\_Y))  
107 self.BTN\_VERSION.isTransparent=**False**108 self.BTN\_VERSION.show()  
109 self.SCREEN.blit(self.IC\_LOGOTEAM, (self.GAME\_WIDTH - 100\*self.SCALE\_X,  
 self.GAME\_HEIGHT - 100\*self.SCALE\_Y))  
110  
111 **def** load\_setting\_pref(self):  
112 **try**:  
113 f = open(self.SETTING\_PREF, **"rt"**)  
114 **except** FileNotFoundError:  
115 *# reset the default pref setting file*116 fx = open(self.SETTING\_PREF, **"w"**)  
117 fx.write(  
118 **"# The map are 0 and 1\ndefault\_map=0\n\n"** \  
 **"# The racers are: rc\_lead, rc\_catus, rc\_turtle, rc\_snail\n"** \  
 **"default\_racer=rc\_snail\ncount\_account=100023\ndefault\_sound=7"**)  
119 fx.close()  
120 self.DEFAULT\_MAP\_CODE = 0  
121 self.DEFAULT\_RACERS\_CODE = **"rc\_snail"**122 self.DEFAULT\_ACCOUNT\_POS = 100000  
123 self.DEFAULT\_SOUND\_CODE = (1, 1, 1)  
124 **return** -1  
125  
126 data = f.readlines()  
127 **for** pref **in** data:  
128 pos = 0  
129 **try**:  
130 **if** pref[0]==**'#'**: *# skip line*131 **continue**132 pos = pref.index(**'='**)  
133 **if** pref[:pos]==**"default\_map"**:  
134 **if** len(pref[pos+1:]) > 0:  
135 self.DEFAULT\_MAP\_CODE = str(int(pref[pos+1:]))  
136 self.assign\_map()  
137 **if** pref[:pos]==**"default\_racer"**:  
138 **if** len(pref[pos+1:]) > 0:  
139 buff = pref[pos+1:]  
140 **while** buff[len(buff)-1]==**'\n'**:  
141 buff = buff[:len(buff)-1]  
142 self.DEFAULT\_RACERS\_CODE = buff  
143 **if** pref[:pos]==**"count\_account"**:  
144 **if** len(pref[pos+1:]) > 0:  
145 buff = pref[pos+1:]  
146 **while** buff[len(buff)-1]==**'\n'**:  
147 buff = buff[:len(buff)-1]  
148 self.DEFAULT\_ACCOUNT\_POS = buff  
149 **if** pref[:pos]==**"default\_sound"**:  
150 **if** len(pref[pos+1:]) > 0:  
151 buff = pref[pos+1:]  
152 **while** buff[len(buff)-1]==**'\n'**:  
153 buff = buff[:len(buff)-1]  
154 num = int(buff)  
155 list = [4, 2, 1]  
156 list2 = []  
157 **for** i **in** range(0,3):  
158 **if** num//list[i]>=1:  
159 list2.append(1)  
160 num = num%list[i]  
161 **else**:  
162 list2.append(0)  
163 self.DEFAULT\_SOUND\_CODE = (list2[0], list2[1], list2[2])  
164 **except** ValueError **as** e:  
165 print(**"Error in file %s"** % \_\_file\_\_)  
166 print(**"Message: %s"** % e)  
167 f.close()  
168 **def** update\_setting\_pref(self):  
169 **try**:  
170 f = open(self.SETTING\_PREF, **"rt"**)  
171 **except** FileNotFoundError:  
172 *# reset the default pref setting file*173 fx = open(self.SETTING\_PREF, **"w"**)  
174 fx.write(**"# The map are 0 and 1\ndefault\_map="**+str(int(self.DEFAULT\_MAP\_CODE))  
175 +**"\n\n# The racers are: rc\_lead, rc\_catus, rc\_turtle, rc\_snail\n"** \  
 **"default\_racer="** + str(self.DEFAULT\_RACERS\_CODE) +**"\n"** \  
176 **"count\_account="**+str(self.DEFAULT\_ACCOUNT\_POS)+**"default\_sound=7\n"**)  
177 fx.close()  
178 **return** -1  
179 data = f.readlines()  
180 buffer = **""**181 **for** pref **in** data:  
182 **try**:  
183 **if** (pref[0]==**'#'**) **or** (pref[0]==**'\n'**): *# skip line*184 buffer+=pref  
185 **continue**186 pos = pref.index(**'='**)  
187 **if** pref[:pos]==**"default\_map"**:  
188 buffer+=**"default\_map="**+str(int(self.DEFAULT\_MAP\_CODE))+**"\n"**189 **continue**190 **if** pref[:pos]==**"default\_racer"**:  
191 buffer += **"default\_racer="** + str(self.DEFAULT\_RACERS\_CODE)+**"\n"**192 **continue**193 **if** pref[:pos]==**"count\_account"**:  
194 buffer += **"count\_account="** + str(self.DEFAULT\_ACCOUNT\_POS)+**"\n"**195 **continue**196 **if** pref[:pos]==**"default\_sound"**:  
197 num = 0  
198 list = [4, 2, 1]  
199 **for** i **in** range(0,3):  
200 num += self.DEFAULT\_SOUND\_CODE[i]\*list[i]  
201 buffer += **"default\_sound="** + str(num)+**"\n"**202 **continue**203  
204  
205 **except** ValueError **as** e:  
206 print(**"Error in file %s"** % \_\_file\_\_)  
207 print(**"Message: %s"** % e)  
208 f.close()  
209 f = open(self.SETTING\_PREF, **"w"**)  
210 f.write(buffer)  
211 f.close()  
212  
213 **def** assign\_map(self):  
214 self.BG\_MAP = self.load\_img(**"img/Background"**+ str(int(self.DEFAULT\_MAP\_CODE))+ **".png"**,  
 self.GAME\_WIDTH\_DEFAULT, self.GAME\_HEIGHT\_DEFAULT)  
215 **def** assign\_racers(self):  
216 space = int(60\*self.GAME\_HEIGHT/self.GAME\_HEIGHT\_DEFAULT)  
217 start\_space = int(self.GAME\_HEIGHT\*260/self.GAME\_HEIGHT\_DEFAULT)  
218 racers = []  
219 **for** i **in** range(0,6):  
220 racers.append(Racer(self.START\_POS, start\_space + i\*space, self,  
 str(self.DEFAULT\_RACERS\_CODE), i))  
221 **return** racers  
222 **pass**223  
224 **class** Amulet(pygame.sprite.Sprite):  
225 **def** \_\_init\_\_(self,game, x, y, kind):  
226 self.x = x  
227 self.y = y  
228 self.kind = kind  
229 self.GAME = game  
230  
231 **def** stop\_amulet(self): *# bau dung*232 self.speed = 0  
233  
234 **def** slow\_amulet(self): *# bua giam toc*235 self.speed=0.75  
236  
237 **def** fast\_amulet(self): *# bua tang toc*238 **if** self.speed + 2 <= 5:  
239 self.speed += 2  
240 self.time -= 2  
241  
242 **def** return\_start\_amulet(self): *# Bua quay ve vi tri xuat phat*243 self.x = self.GAME.START\_POS  
244 self.time = 0  
245  
246 **def** teleport\_amulet(self): *# Bua toc bien*247 self.x += 200  
248 self.time = 0  
249  
250 **def** win\_amulet(self): *# Bua bay thang den dich*251 self.x = self.GAME.DISTANCE  
252 self.time = 0  
253  
254 **def** turnback\_amulet(self): *# Bua quay dau*255 self.speed = -abs(self.speed)  
256 self.time -= 2  
257 self.exist\_turn = **True**258  
259 **def** active(self):  
260 **if**(self.kind == 1):  
261 **if**(self.exist\_shield\_amulet != **True**):  
262 self.stop\_amulet()  
263 **else**:  
264 self.kind = 0  
265 self.exist\_shield\_amulet = **False**266 **elif**(self.kind == 2):  
267 **if** (self.exist\_shield\_amulet != **True**):  
268 self.slow\_amulet()  
269 **else**:  
270 self.kind=0  
271 self.exist\_shield\_amulet = **False**272 **elif** self.kind == 3:  
273 **if** self.exist\_shield\_amulet != **True**:  
274 self.fast\_amulet()  
275 **else**:  
276 self.kind = 0  
277 self.exist\_shield\_amulet = **False**278 **elif** self.kind == 4:  
279 **if** self.exist\_shield\_amulet != **True**:  
280 self.return\_start\_amulet()  
281 **else**:  
282 self.kind = 0  
283 self.exist\_shield\_amulet = **False**284 **elif** self.kind == 5:  
285 **if** (self.exist\_shield\_amulet != **True**):  
286 self.teleport\_amulet()  
287 **else**:  
288 self.kind = 0  
289 self.exist\_shield\_amulet = **False**290 **elif** (self.kind == 6):  
291 **if** (self.exist\_shield\_amulet != **True**):  
292 self.win\_amulet()  
293 **else**:  
294 self.kind = 0  
295 self.exist\_shield\_amulet = **False**296 **elif** (self.kind == 7):  
297 **if** (self.exist\_shield\_amulet != **True**):  
298 self.turnback\_amulet()  
299 **else**:  
300 self.kind = 0  
301 self.exist\_shield\_amulet = **False**302 self.exist\_amulet = **False**303  
304 **def** Amulet\_appear(self):  
305 object=[**True**, **True**, **True**, **False**, **False**]  
306 list\_amulet=[1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,4,5,5,5,7,7,7,6]  
307 self.exist\_amulet=random.choice(object)  
308 self.amulet\_x = self.x + random.randrange(300, 700)  
309 **if** self.amulet\_x > self.GAME.DISTANCE-500:  
310 self.exist\_amulet = **False**311 **if** self.exist\_amulet == **True**:  
312 self.kind = random.choice(list\_amulet)  
313 self.time=100  
314 **else**:  
315 self.kind=0  
316  
317 **def** draw\_amulet(self, rollback):  
318 **if** self.exist\_amulet:  
319 **if** (self.kind == 1):  
320 self.GAME.SCREEN.blit(self.IC\_STOP, (self.amulet\_x + rollback, self.y))  
321 **elif** (self.kind == 2):  
322 self.GAME.SCREEN.blit(self.IC\_SLOW, (self.amulet\_x + rollback, self.y))  
323 **elif** (self.kind == 3):  
324 self.GAME.SCREEN.blit(self.IC\_FAST, (self.amulet\_x + rollback, self.y))  
325 **elif** (self.kind == 4):  
326 self.GAME.SCREEN.blit(self.IC\_RETURN\_START, (self.amulet\_x + rollback, self.y))  
327 **elif** (self.kind == 5):  
328 self.GAME.SCREEN.blit(self.IC\_TELEPORT, (self.amulet\_x + rollback, self.y))  
329 **elif** (self.kind == 6):  
330 self.GAME.SCREEN.blit(self.IC\_WIN, (self.amulet\_x + rollback, self.y))  
331 **elif** (self.kind == 7):  
332 self.GAME.SCREEN.blit(self.IC\_TURNBACK, (self.amulet\_x + rollback, self.y))  
333 self.amulet\_x += +rollback  
334  
335 **class** Racer(Amulet): *# Doi tuong dua*336 **def** \_\_init\_\_(self, x, y, game, pack=**"rc\_snail"**, num=**"0"**):  
337 self.pack\_sprite = pack  
338 self.num = num  
339 name = **"img/"**340 self.ic\_name = name + pack + str(num) + **".png"**341 self.img = game.load\_img(self.ic\_name, -1, 48)  
342 self.imgbig = game.load\_img(self.ic\_name, -1, 48\*2)  
343 ic\_name\_turn = name + pack + str(num) + str(num) + **".png"**344 self.img\_turn = game.load\_img(ic\_name\_turn, -1, 48)  
345 self.x = x  
346 self.y = y  
347 self.y\_def = y  
348 self.t\_jump = 0  
349 self.stun = 0  
350 self.time = 0  
351 self.speed = random.randrange(15, 30)/10  
352 self.rank = num + 1  
353 self.GAME = game  
354 self.distance = 0  
355 self.amulet\_x = 0  
356 self.IC\_STOP = game.load\_img(**"img/ic\_amulet"** + **"1"** + **".png"**, 55, 57)  
357 self.IC\_SLOW = game.load\_img(**"img/ic\_amulet"** + **"2"** + **".png"**, 55, 57)  
358 self.IC\_FAST = game.load\_img(**"img/ic\_amulet"** + **"3"** + **".png"**, 55, 57)  
359 self.IC\_RETURN\_START = game.load\_img(**"img/ic\_amulet"** + **"4"** + **".png"**, 55, 57)  
360 self.IC\_TELEPORT = game.load\_img(**"img/ic\_amulet"** + **"5"** + **".png"**, 55, 57)  
361 self.IC\_WIN = game.load\_img(**"img/ic\_amulet"** + **"6"** + **".png"**, 55, 57)  
362 self.IC\_TURNBACK = game.load\_img(**"img/ic\_amulet"** + **"7"** + **".png"**, 55, 57)  
363 self.IC\_SHIELD = game.load\_img(**"img/ic\_shield.png"**, 60, 60)  
364 self.img\_protect= game.load\_img(**"img/ic\_protect.png"**, 100, 100)  
365 self.IC\_STAR = game.load\_img(**"img/ic\_star.png"**, 80, 80)  
366 self.item\_star\_amulet = 1  
367 self.item\_shield\_amulet = 1  
368 self.kind=0  
369 self.exist\_shield\_amulet = **False**370 self.exist\_star\_amulet = **False**371 self.exist\_amulet = **False**372 self.exist\_turn = **False**373 self.button\_shield\_amulet = **True**374 self.pos\_rank = self.rank  
375  
376 **def** update(self,camera):  
377 **if** self.exist\_turn==**True** :  
378 self.GAME.SCREEN.blit(self.img\_turn, (self.x, self.y))  
379 **elif**(self.exist\_shield\_amulet):  
380 self.GAME.SCREEN.blit(self.img, (self.x, self.y))  
381 self.GAME.SCREEN.blit(self.img\_protect,  
 (self.x - 23\*self.GAME.SCALE\_X, self.y - 22\*self.GAME.SCALE\_Y))  
382 **else**:  
383 self.GAME.SCREEN.blit(self.img, (self.x, self.y))  
384 **if** self.pos\_rank < self.rank:  
385 self.pos\_rank += 0.25  
386 **if** self.pos\_rank > self.rank:  
387 self.pos\_rank -= 0.25  
388 **if** self.x +self.speed + camera.delta >= self.GAME.DISTANCE:  
389 self.x = self.GAME.DISTANCE  
390 self.speed = 0  
391 **if** self.rank == 1:  
392 **if** self.y <= self.y\_def:  
393 self.y = self.y\_def - self.t\_jump\*self.GAME.GAME\_HEIGHT/20\  
 + 9.8\*self.t\_jump\*self.t\_jump/2  
394 self.t\_jump += 0.25  
395 **else**:  
396 self.t\_jump = 0  
397 self.y = self.y\_def  
398 **return False**399 **else**:  
400 **if** self.x + self.speed\*self.GAME.SCALE\_X + camera.delta <= self.GAME.DISTANCE:  
401 self.x += self.speed\*self.GAME.SCALE\_X + camera.delta  
402 **else**:  
403 self.x = self.GAME.DISTANCE  
404 **return True**405 **def** updatespeed(self):  
406 self.speed = random.randrange(15, 30)/10  
407  
408 **class** History():  
409 **def** \_\_init\_\_(self):  
410 self.racerType = **""**411 self.racerNum = **"0"**412 self.coinResult = **""**413 **pass**414  
415  
416 **class** Ranking(): *#Bang xep hang*417 **def** \_\_init\_\_(self, game, rs):  
418 self.GAME = game  
419 self.show\_top1 = **False**420 self.img = self.GAME.IC\_RANK  
421 self.size = self.img.get\_rect().size  
422 self.img = pygame.transform.scale(self.img, (int(self.size[0]\*(game.GAME\_WIDTH/1280)),  
 int(self.size[1]\*(game.GAME\_HEIGHT/720))))  
423 self.size = self.img.get\_rect().size  
424 self.x = game.GAME\_WIDTH - self.size[0]  
425 self.y = 0  
426  
427 **def** draw(self, rs):  
428 m = self.x + int(self.size[0]/2.5)  
429 n = int(self.size[1]/7.7)  
430 self.GAME.SCREEN.blit(self.img, (self.x, self.y))  
431 **for** i **in** range(0, 6):  
432 self.GAME.SCREEN.blit(rs[i].img, (m,self.y + n\*(rs[i].pos\_rank+0.33)))  
433 **if** self.show\_top1:  
434 self.GAME.SCREEN.blit(self.GAME.IC\_TOP1, (m, self.y + n))  
435  
436 **def** update(self, rs):  
437 **for** i **in** range(0, 6):  
438 **for** j **in** range(0, 6):  
439 **if** rs[i].x > rs[j].x **and** rs[i].rank > rs[j].rank:  
440 rs[i].rank, rs[j].rank = rs[j].rank, rs[i].rank  
441 self.draw(rs)  
442  
443  
444 **class** User():  
445 **def** \_\_init\_\_(self, coin = **"1"**):  
446 self.ID = 0  
447 self.name = **"NULL"**448 self.password = **"NULL"**449 self.winrate = **"0"**450 self.playTime = **"0"**451 self.coins = coin  
452 self.item\_shield = 0  
453 self.use\_star=**False**454 **def** cloneTo(self,temp\_user):  
455 temp\_user.ID = self.ID  
456 temp\_user.name = self.name  
457 temp\_user.password = self.password  
458 temp\_user.winrate = self.winrate  
459 temp\_user.playTime = self.playTime  
460 temp\_user.coins = self.coins  
461 temp\_user.item\_shield = self.item\_shield  
462 **pass**463  
464 **class** Camera():  
465 **def** \_\_init\_\_(self, game):  
466 self.GAME = game  
467 self.follow = 3  
468 self.x = game.GAME\_WIDTH/3  
469 self.delta = 0  
470 self.anim = 0  
471 self.time = 0  
472  
473 **def** update(self, rs):  
474 delta = self.x - rs.x  
475 **if** int(self.time % 10) == 0:  
476 self.anim = (self.anim + 1) % 3  
477 self.time =0  
478 self.time += 1  
479 self.GAME.SCREEN.blit(self.GAME.IC\_CAMERA[self.anim], (rs.x, rs.y-20))  
480 **if** abs(delta) < 2\*abs(rs.speed\*self.GAME.SCALE\_X):  
481 self.delta = rs.x - self.x  
482 **else**:  
483 **if** delta == 0:  
484 self.delta = 0  
485 **else**:  
486 self.delta = rs.speed\*(delta/abs(delta))\*self.GAME.SCALE\_X  
487 **if** abs(delta) > 80:  
488 self.delta = max(abs(delta)/3, 25)\*(delta/abs(delta))  
489  
490 *# change following*491 **if** pygame.key.get\_pressed()[pygame.K\_1]:  
492 self.follow = 0  
493 **return**494 **if** pygame.key.get\_pressed()[pygame.K\_2]:  
495 self.follow = 1  
496 **return**497 **if** pygame.key.get\_pressed()[pygame.K\_3]:  
498 self.follow = 2  
499 **return**500 **if** pygame.key.get\_pressed()[pygame.K\_4]:  
501 self.follow = 3  
502 **return**503 **if** pygame.key.get\_pressed()[pygame.K\_5]:  
504 self.follow = 4  
505 **return**506 **if** pygame.key.get\_pressed()[pygame.K\_6]:  
507 self.follow = 5  
508 **return**509 **pass**510  
511 **class** Minimap():  
512 **def** \_\_init\_\_(self, game):  
513 self.GAME = game  
514 self.img\_minimap = game.IC\_MINIMAP  
515 self.size = self.img\_minimap.get\_rect().size  
516 self.img\_b\_point = game.IC\_POINT\_B  
517 self.size\_p = self.img\_b\_point.get\_rect().size  
518 self.img\_r\_point = game.IC\_POINT\_R  
519 self.img\_camera = game.IC\_MINIMAP\_CAMERA  
520 self.size\_c = self.img\_camera.get\_rect().size  
521 self.x = game.GAME\_WIDTH/2 - self.size[0]/2  
522 self.y = game.GAME\_HEIGHT/10\*9  
523  
524 **def** update(self, racers, play\_choose, camera\_follow):  
525 self.GAME.SCREEN.blit(self.img\_minimap, (self.x, self.y))  
526 a = self.GAME.GAME\_WIDTH/3  
527 **for** r **in** racers:  
528 **if** r.num != play\_choose:  
529 x = self.x + self.size[0]\*(1 - (self.GAME.DISTANCE - r.x)/ \  
530 (self.GAME.DISTANCE\_DEFAULT - a)) - self.size\_p[0]/2  
531 y = self.y + self.size[1]/2  
532 self.GAME.SCREEN.blit(self.img\_b\_point, (x, y))  
533 **for** r **in** racers:  
534 **if** r.num == play\_choose:  
535 x = self.x + self.size[0]\*(1 - (self.GAME.DISTANCE - r.x)/ \  
536 (self.GAME.DISTANCE\_DEFAULT - a)) - self.size\_p[0]/2  
537 y = self.y + self.size[1]/2  
538 self.GAME.SCREEN.blit(self.img\_r\_point, (x, y))  
539 **for** r **in** racers:  
540 **if** r.num == camera\_follow:  
541 x = self.x + self.size[0]\*(1 - (self.GAME.DISTANCE - r.x)/  
542 (self.GAME.DISTANCE\_DEFAULT-a))-self.size\_p[0]/2-self.size\_c[0]/2+self.size\_p[0]/2  
543 y = self.y + self.size[1]/2 - self.size\_c[1]/2 + self.size\_p[1]/2  
544 self.GAME.SCREEN.blit(self.img\_camera, (x, y))  
545 **pass**546

*File Pages.py:* giao diện các trang

001 **from** libs.global\_variables **import** \*  
002 **from** libs.Widgets **import** \*  
003   
004 **import** pygame  
005   
006 **class** LoginPage():  
007 **def** \_\_init\_\_(self, gameLancher):  
008 *#load gameplay from main*009 self.GAME = gameLancher  
010 self.SCREEN = pygame.display.get\_surface()  
011 self.loginForm = self.GAME.load\_img(**"img/pg\_mainpage\_no\_title.png"**,  
 -1, self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
012 self.rect = self.loginForm.get\_rect()  
013 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
014 *#add text and input*015 self.TITLE = View(self.rect.x + self.rect.w\*0.5, self.rect.y  
 + self.rect.h\*0.067, text=**"LOG IN/SIGN UP"**, gravity=**"center"**)  
016 self.userNameInput = InputBox(self.rect.x +self.rect.w\*0.4, self.rect.y  
 + self.rect.h\*0.274, 250\*self.GAME.SCALE\_X, 50\*self.GAME.SCALE\_Y)  
017 self.passWordInput = InputBox(self.rect.x +self.rect.w\*0.4, self.rect.y  
 + self.rect.h\*0.467, 250\*self.GAME.SCALE\_X, 50\*self.GAME.SCALE\_Y)  
018 self.userNameText = View(self.rect.x +self.rect.w\*0.2, self.rect.y  
 + self.rect.h\*0.274, text=**"Username: "**, gravity=**"center\_left"**)  
019 self.passWordText = View(self.rect.x +self.rect.w\*0.2, self.rect.y  
 + self.rect.h\*0.467, text=**"Password: "**, gravity=**"center\_left"**)  
020 self.btn\_signin = View(self.rect.x +self.rect.w\*0.2234,  
 self.rect.y + self.rect.h\*0.86, **"Sign in"**, gravity=**"center"**)  
021 self.btn\_signup = View(self.rect.x +self.rect.w\*0.745,  
 self.rect.y + self.rect.h\*0.86, **"Sign up"**, gravity=**"center"**)  
022 self.warningText = View(self.rect.x +self.rect.w\*0.511,  
 self.rect.y + self.rect.h\*0.7, **""**)  
023 self.showView = [self.TITLE, self.userNameText, self.passWordText,  
 self.btn\_signup, self.btn\_signin, self.warningText]  
024 **pass**025 **def** drawLoginPage(self):  
026 self.SCREEN.blit(self.loginForm, self.rect)  
027 self.SCREEN.blit(self.GAME.IC\_APP\_NAME, (self.rect.x + self.rect.w//2  
 - self.GAME.IC\_APP\_NAME.get\_rect().w//2, 100\*self.GAME.SCALE\_Y))  
028 self.userNameInput.draw(self.SCREEN)  
029 self.passWordInput.draw(self.SCREEN)  
030 **for** v **in** self.showView:  
031 v.show()  
032 **pass**033 **def** loginHandle(self, event):  
034 ev\_name = self.userNameInput.handle\_event(event)  
035 **if** ev\_name == 0:  
036 self.warningText.setText(**"Username is not longer than 10 character"**)  
037 **elif** ev\_name == 1:  
038 self.warningText.setText(**""**)  
039 ev\_pass = self.passWordInput.handle\_event(event)  
040 **if** ev\_pass == 0:  
041 self.warningText.setText(**"Password is not longer than 10 character"**)  
042 **elif** ev\_pass == 1:  
043 self.warningText.setText(**""**)  
044   
045 **class** MainPage():  
046 **def** \_\_init\_\_(self, gameLancher):  
047 *#load screen from main*048 self.GAME = gameLancher  
049 self.SCREEN = pygame.display.get\_surface()  
050 self.mainForm = gameLancher.load\_img(**"img/pg\_mainpage\_no\_title.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//4, self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
051 self.rect = self.mainForm.get\_rect()  
052 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
053 self.reloadBtn()  
054 **pass**055   
056 **def** reloadBtn(self):  
057 self.TITLE = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.067, text=**"MAIN MENU"**, gravity=**"center"**)  
058 self.btn\_start = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.252, **"START"**, gravity=**"center"**)  
059 self.btn\_setting = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.412, **"SETTING"**, gravity=**"center"**)  
060 self.btn\_history = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.57, **"HISTORY"**, gravity=**"center"**)  
061 self.btn\_store = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.73, **"SHOP"**, gravity=**"center"**)  
062 self.btn\_logout = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.9, **"LOG OUT"**, gravity=**"center"**)  
063 self.button = (self.btn\_start, self.btn\_setting,  
 self.btn\_store, self.btn\_logout, self.btn\_history)  
064   
065 **def** drawMainPage(self):  
066 self.SCREEN.blit(self.mainForm, self.rect)  
067 self.TITLE.show()  
068 **for** btn **in** self.button:  
069 btn.show()  
070 **pass**071   
072 **def** drawInitStart(self, user, racers):  
073 self.mainForm = self.GAME.load\_img(**"img/pg\_mainpage\_no\_title.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//3\*2, self.GAME.GAME\_WIDTH\_DEFAULT//2)  
074 self.rect = self.mainForm.get\_rect()  
075 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
076   
077 TITLE = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.05, text=**"CONFIGURING INIT"**, gravity=**"center"**)  
078 btn\_save = View(self.rect.x + self.rect.w\*0.25, self.rect.y+self.rect.h  
 - 20\*self.GAME.SCALE\_Y, text=**"LET'S GO!"**, gravity=**"mid\_bottom"**)  
079 btn\_back = View(self.rect.x + self.rect.w\*0.738, self.rect.y+self.rect.h  
 - 20\*self.GAME.SCALE\_Y, text=**"BACK!"**, gravity=**"mid\_bottom"**)  
080 tv\_coin = View(self.rect.x + self.rect.w\*0.428,  
 self.rect.y + self.rect.h\*0.225, text=**"Enter coins:"**, gravity=**"mid\_left"**)  
081 tv\_distance = View(self.rect.x + self.rect.w\*0.428,  
 self.rect.y + self.rect.h\*0.35, text=**"Range of map:"**, gravity=**"mid\_left"**)  
082 ip\_coin = InputBox(self.rect.x + self.rect.w\*0.77 - 100\*self.GAME.SCALE\_X,  
 self.rect.y + self.rect.h\*0.225 - 25\*self.GAME.SCALE\_Y,  
083 200\*self.GAME.SCALE\_X, 50\*self.GAME.SCALE\_Y, **""**, isdigit=**True**)  
084 ip\_distance = InputBox(self.rect.x + self.rect.w\*0.77 - 100\*self.GAME.SCALE\_X,  
 self.rect.y + self.rect.h\*0.35 - 25\*self.GAME.SCALE\_Y,  
085 200\*self.GAME.SCALE\_X, 50\*self.GAME.SCALE\_Y, **""**, isdigit=**True**)  
086 self.warrningText = View(self.rect.x + self.rect.w\*0.665,  
 self.rect.y + self.rect.h\*0.457, **""**, gravity=**"center"**)  
087 self.warrningText2 = View(self.rect.x + self.rect.w\*0.573,  
 self.rect.y + self.rect.h\*0.842, **""**, gravity=**"center"**)  
088 tv\_star = View(self.rect.x + self.rect.w\*0.486,  
 self.rect.y + self.rect.h\*0.758, text=**"Star Amulet"**, gravity=**"center"**)  
089 tv\_shield = View(self.rect.x + self.rect.w\*0.79,  
 self.rect.y + self.rect.h\*0.758, text=**"Shield"**, gravity=**"center"**)  
090 list\_tv = [tv\_coin, tv\_distance, self.warrningText,  
 self.warrningText2, tv\_shield, tv\_star, TITLE, btn\_save, btn\_back]  
091 list\_ip = [ip\_coin, ip\_distance]  
092   
093 list\_imgRacer = []  
094 imgArrow = self.GAME.IC\_ARROW  
095 imgTick = self.GAME.IC\_TICK  
096   
097 imgStar = ImageView(self.GAME, self.rect.x + self.rect.w\*0.428,  
 self.rect.y + self.rect.h\*0.522, 100, 100, **"img/ic\_star.png"**)  
098 imgShiled = ImageView(self.GAME, self.rect.x + self.rect.w\*0.721,  
 self.rect.y + self.rect.h\*0.522, 100, 100, **"img/ic\_shield.png"**)  
099 list\_imgExtend = [imgStar, imgShiled]  
100 use\_star = **False**101 use\_shield = **False**102   
103 **for** i **in** range(0, 6):  
104 list\_imgRacer.append(ImageView(self.GAME, self.rect.x+self.rect.w\*0.075,  
 self.rect.y + self.rect.h\*0.215 + (racers[i].img.get\_rect().h+14)\*i,  
 -1, 60, racers[i].ic\_name))  
105 list\_imgRacer[0].setActive(**True**)  
106 active\_pos = 0  
107   
108 **while True**:  
109 self.SCREEN.blit(self.mainForm, self.rect)  
110 self.SCREEN.blit(imgArrow, (self.rect.x+self.rect.w\*0.075+  
 list\_imgRacer[active\_pos].rect.w + 5, self.rect.y+self.rect.h\*0.215  
 +(list\_imgRacer[active\_pos].rect.h+3)\*active\_pos))  
111 **for** tv **in** list\_tv:  
112 tv.show()  
113 **for** imgE **in** list\_imgExtend:  
114 imgE.draw(self.SCREEN)  
115 use\_star **and** self.SCREEN.blit(imgTick,  
 (self.rect.x + self.rect.w\*0.54, self.rect.y + self.rect.h\*0.509))  
116 use\_shield **and** self.SCREEN.blit(imgTick,  
 (self.rect.x + self.rect.w\*0.831, self.rect.y + self.rect.h\*0.509))  
117 **for** ip **in** list\_ip:  
118 ip.draw(self.SCREEN)  
119 **for** rc **in** list\_imgRacer:  
120 rc.draw(self.SCREEN)  
121 **if** rc.is\_clicked():  
122 list\_imgRacer[active\_pos].setActive(**False**)  
123 active\_pos = list\_imgRacer.index(rc)  
124 rc.setActive(**True**)  
125 **if** imgStar.is\_clicked():  
126 **if not** use\_star **and** len(str(ip\_coin.text)) > 0 \  
 **and** int(ip\_coin.text) >= int(user.coins)/2:  
127 use\_star = **not** use\_star  
128 imgStar.setActive(use\_star)  
129 self.warrningText2.setText(**"Important: You've choose Star Amulet"**)  
130 **elif not** use\_star:  
131 self.warrningText2.setText(**"Can't choose while your bets "** +  
 **"half less than your coins"**)  
132 **else**:  
133 use\_star = **not** use\_star  
134 imgStar.setActive(use\_star)  
135 self.warrningText2.setText(**""**)  
136 time.sleep(0.1)  
137 **if** imgShiled.is\_clicked():  
138 **if** user.item\_shield > 0:  
139 use\_shield = **not** use\_shield  
140 imgShiled.setActive(use\_shield)  
141 **else**:  
142 self.warrningText2.setText(**"You need buy shield in shop"**)  
143 time.sleep(0.1)  
144 **if** use\_star **and** len(str(ip\_coin.text)) > 0 \  
 **and** int(ip\_coin.text) < int(user.coins)/2:  
145 self.warrningText2.setText(**"Can't chosse while"** +  
 **" your bets half less than your coins"**)  
146 use\_star = **False**147 imgStar.setActive(use\_star)  
148 **if** btn\_back.is\_clicked(gameLancher=self.GAME):  
149 self.mainForm = self.GAME.load\_img(**"img/pg\_mainpage\_no\_title.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//4, self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
150 self.rect = self.mainForm.get\_rect()  
151 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
152 **return False**, **None**, **None**, active\_pos, **False**, **False**153 **if** btn\_save.is\_clicked(gameLancher=self.GAME):  
154 **if** len((list\_ip[0].text))==0 **or** len((list\_ip[1].text))==0:  
155 self.warrningText.setText(**"Please enter data to the box"**)  
156 **elif** int(list\_ip[0].text) > int(user.coins):  
157 self.warrningText.setText(**"You've not enough coins"**)  
158 **elif** int(list\_ip[1].text) > 9000 **or** int(list\_ip[1].text) < 2500:  
159 self.warrningText.setText(**"Please enter distance between 2500 and 9000"**)  
160 **else**:  
161 self.mainForm = self.GAME.load\_img(**"img/pg\_mainpage\_no\_title.png"**, \  
 self.GAME.GAME\_WIDTH\_DEFAULT//4, self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
162 self.rect = self.mainForm.get\_rect()  
163 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
164 *# return coin - distance*165 **return True**, list\_ip[0].text, int(list\_ip[1].text),\  
 active\_pos, use\_star, use\_shield  
166   
167 **for** event **in** pygame.event.get():  
168 **if** event.type == pygame.QUIT:  
169 pygame.quit()  
170 exit(0)  
171 **for** box **in** list\_ip:  
172 box.handle\_event(event)  
173 pygame.display.flip()  
174 **class** SettingPage():  
175 **def** \_\_init\_\_(self, gameLancher):  
176 self.GAME = gameLancher  
177 self.SCREEN = pygame.display.get\_surface()  
178 self.settingForm = self.GAME.load\_img(**"img/pg\_mainpage\_no\_title.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//2, self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
179 self.rect = self.settingForm.get\_rect()  
180 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
181   
182 self.TITLE = View(self.rect.x +self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.0535, text=**"SETTINGS"**, gravity=**"center"**)  
183 self.btn\_setplayer = View(self.rect.x +self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.217, text=**"Choose set of player"**, gravity=**"center"**)  
184 self.btn\_setmap = View(self.rect.x +self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.4, text=**"Choose set of map"**, gravity=**"center"**)  
185 self.btn\_modsound = View(self.rect.x +self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.625, text=**"Sounds"**, gravity=**"center"**)  
186 self.btn\_back = View(self.rect.x +self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.8226, text=**"Back"**, gravity=**"center"**)  
187 self.button = (self.btn\_setplayer, self.btn\_setmap,  
 self.btn\_back, self.TITLE, self.btn\_modsound)  
188   
189 **def** drawSettingPage(self):  
190 self.TITLE.setText(**"SETTINGS"**)  
191 self.SCREEN.blit(self.settingForm, self.rect)  
192 **for** btn **in** self.button:  
193 btn.show()  
194 **pass**195 **def** drawChooseRacer(self):  
196 self.TITLE.setText(**"RACER CHOOSER"**)  
197 self.LIST\_RC = []  
198 self.listRacer = [**"rc\_turtle"**, **"rc\_lead"**, **"rc\_snail"**, **"rc\_parrot"**, **"rc\_player"**]  
199 last\_active = 0  
200 btn\_save = View(self.rect.x + self.rect.w//2, self.rect.y + self.rect.h  
 - 30\*self.GAME.SCALE\_X, text=**"SAVE"**, gravity=**"center"**)  
201 **for** i **in** range(0, len(self.listRacer)):  
202 self.LIST\_RC.append(ImageView(self.GAME,  
 150\*self.GAME.SCALE\_X\*(i % 3)+ self.GAME.GAME\_WIDTH/3,  
 100\*self.GAME.SCALE\_Y\*(i//3) + self.GAME.GAME\_HEIGHT/3, 60\*self.GAME.SCALE\_X,  
 60\*self.GAME.SCALE\_Y, **"img/"** + self.listRacer[i] + **"1.png"**))  
203 **if** self.listRacer[i] == (self.GAME.DEFAULT\_RACERS\_CODE):  
204 self.LIST\_RC[i].setActive(**True**)  
205 last\_active = i  
206 **while True**:  
207 self.SCREEN.blit(self.settingForm, self.rect)  
208 **for** rc **in** self.LIST\_RC:  
209 rc.draw(self.SCREEN)  
210 **if** rc.is\_clicked():  
211 self.LIST\_RC[last\_active].setActive(**False**)  
212 last\_active = self.LIST\_RC.index(rc)  
213 self.LIST\_RC[last\_active].setActive(**True**)  
214 **for** event **in** pygame.event.get():  
215 **if** event.type == pygame.QUIT:  
216 pygame.quit()  
217 exit(0)  
218 self.TITLE.show()  
219 btn\_save.show()  
220 **if** btn\_save.is\_clicked(gameLancher=self.GAME):  
221 **return** self.listRacer[last\_active]  
222 pygame.display.flip()  
223 **def** drawChooseMap(self):  
224 self.TITLE.setText(**"MAPS CHOOSER"**)  
225 self.LIST\_BG = []  
226 last\_active = 0  
227 btn\_save = View(self.rect.x + self.rect.w//2, self.rect.y +\  
 self.rect.h - 30\*self.GAME.SCALE\_Y,text=**"SAVE"**, gravity=**"center"**)  
228 img\_info = []  
229 *# create table of maps element*230 row = 2  
231 collum = 2  
232 **for** i **in** range(0, row):  
233 **for** j **in** range(0, collum):  
234 fname = **"img/Background"**+str(i\*2+j)+**".png"**235 **try**:  
236 open(fname, **'r'**).close()  
237 self.LIST\_BG.append(ImageView(self.GAME, self.rect.x + self.rect.w\*0.15 +  
 (170 + 60)\*self.GAME.SCALE\_X\*j, self.rect.y + self.rect.h\*0.18 +  
 (95.625+ 35)\*self.GAME.SCALE\_Y \*i, 170, -1, fname))  
238 img\_info.append(View(self.rect.x + self.rect.w\*0.15 + (170+60)\*  
 self.GAME.SCALE\_X\*j + 170/2\*self.GAME.SCALE\_X, self.rect.y  
239 + self.rect.h\*0.18 +(95.625+35)\*self.GAME.SCALE\_Y \*(i+1)  
 - 35/2\*self.GAME.SCALE\_Y, **""**, gravity=**"center"**))  
240 **if** i\*2+j == int(self.GAME.DEFAULT\_MAP\_CODE):  
241 self.LIST\_BG[i\*2+j].setActive(**True**)  
242 last\_active = i\*2+j  
243 **except** FileNotFoundError:  
244 **break**245 img\_info[0].setText(**"Spring"**)  
246 img\_info[1].setText(**"The winter"**)  
247 img\_info[2].setText(**"Summer vacation"**)  
248 img\_info[3].setText(**"The Autumn"**)  
249 **while True**:  
250 self.SCREEN.blit(self.settingForm, self.rect)  
251 **for** info **in** img\_info:  
252 info.show()  
253 **for** bg **in** self.LIST\_BG:  
254 bg.draw(self.SCREEN)  
255 **if** bg.is\_clicked():  
256 self.LIST\_BG[last\_active].setActive(**False**)  
257 last\_active = self.LIST\_BG.index(bg)  
258 self.LIST\_BG[last\_active].setActive(**True**)  
259 **for** event **in** pygame.event.get():  
260 **if** event.type == pygame.QUIT:  
261 pygame.quit()  
262 exit(0)  
263 self.TITLE.show()  
264 btn\_save.show()  
265 **if** btn\_save.is\_clicked(self.GAME):  
266 **return** last\_active  
267 pygame.display.flip()  
268 **def** drawOptionSound(self):  
269 self.TITLE.setText(**"SOUND OPTIONS"**)  
270 btn\_save = View(self.rect.x + self.rect.w//2, self.rect.y  
 + self.rect.h - 30\*self.GAME.SCALE\_Y, text=**"SAVE"**, gravity=**"center"**)  
271 btn\_theme = View(self.rect.x + self.rect.w//2,  
 self.rect.y + self.rect.h\*(1 - 0.6), text=**"THEME SONG:"**, gravity=**"center"**)  
272 btn\_racing = View(self.rect.x + self.rect.w//2,  
 self.rect.y + self.rect.h\*(1 - 0.5), text=**"RACING SONG:"**, gravity=**"center"**)  
273 btn\_effect = View(self.rect.x + self.rect.w//2,  
 self.rect.y + self.rect.h\*(1 - 0.4), text=**"EFFECT SOUND:"**, gravity=**"center"**)  
274 **if** self.GAME.DEFAULT\_SOUND\_CODE[0] == 1:  
275 btn\_theme.text += **" ON"**276 **else**:  
277 btn\_theme.text += **" OFF"**278 **if** self.GAME.DEFAULT\_SOUND\_CODE[1] == 1:  
279 btn\_racing.text += **" ON"**280 **else**:  
281 btn\_racing.text += **" OFF"**282 **if** self.GAME.DEFAULT\_SOUND\_CODE[2] == 1:  
283 btn\_effect.text += **" ON"**284 **else**:  
285 btn\_effect.text += **" OFF"**286 time.sleep(0.1)  
287 **while True**:  
288 self.SCREEN.blit(self.settingForm, self.rect)  
289 **for** event **in** pygame.event.get():  
290 **if** event.type == pygame.QUIT:  
291 pygame.quit()  
292 exit(0)  
293 self.TITLE.show()  
294 btn\_save.show()  
295 btn\_theme.show()  
296 btn\_effect.show()  
297 btn\_racing.show()  
298 **if** btn\_save.is\_clicked(self.GAME):  
299 **return** (btn\_theme.text == **"THEME SONG: ON"**,  
300 btn\_racing.text == **"RACING SONG: ON"**,  
301 btn\_effect.text == **"EFFECT SOUND: ON"**)  
302 **if** btn\_theme.is\_clicked(self.GAME):  
303 **if** btn\_theme.text == **"THEME SONG: ON"**:  
304 btn\_theme.text = **"THEME SONG: OFF"**305 **else**:  
306 btn\_theme.text = **"THEME SONG: ON"**307 time.sleep(0.2)  
308 **if** btn\_racing.is\_clicked(self.GAME):  
309 **if** btn\_racing.text == **"RACING SONG: ON"**:  
310 btn\_racing.text = **"RACING SONG: OFF"**311 **else**:  
312 btn\_racing.text = **"RACING SONG: ON"**313 time.sleep(0.2)  
314 **if** btn\_effect.is\_clicked(self.GAME):  
315 **if** btn\_effect.text == **"EFFECT SOUND: ON"**:  
316 btn\_effect.text = **"EFFECT SOUND: OFF"**317 **else**:  
318 btn\_effect.text = **"EFFECT SOUND: ON"**319 time.sleep(0.2)  
320 pygame.display.flip()  
321   
322 **class** InfoZone():  
323 **def** \_\_init\_\_(self, gameLancher, user):  
324 self.GAME = gameLancher  
325 self.USER = user  
326 self.SCREEN = pygame.display.get\_surface()  
327 self.player\_name = View(60\*self.GAME.SCALE\_X, 35\*self.GAME.SCALE\_Y,  
 text=user.name, gravity=**"center\_horizontal"**)  
328 self.coin = View(60\*self.GAME.SCALE\_X, 80\*self.GAME.SCALE\_Y,  
 text=str(user.coins), gravity=**"center\_horizontal"**)  
329 self.rect = pygame.Rect((10\*self.GAME.SCALE\_X, 10\*self.GAME.SCALE\_Y),  
 (200\*self.GAME.SCALE\_X, 70\*self.GAME.SCALE\_Y))  
330 **pass**331 **def** drawInfoZone(self):  
332 self.SCREEN.blit(self.GAME.IC\_PROFILE, (10\*self.GAME.SCALE\_X, 10\*self.GAME.SCALE\_Y))  
333 self.SCREEN.blit(self.GAME.IC\_COIN, (10\*self.GAME.SCALE\_X, 60\*self.GAME.SCALE\_Y))  
334 self.player\_name.show()  
335 self.coin.show()  
336 **def** drawInfoZoneExpand(self):  
337 self.SCREEN.blit(self.GAME.IC\_PROFILE, (10\*self.GAME.SCALE\_X, 10\*self.GAME.SCALE\_Y))  
338 self.SCREEN.blit(self.GAME.IC\_COIN, (10\*self.GAME.SCALE\_X, 60\*self.GAME.SCALE\_Y))  
339 self.SCREEN.blit(self.GAME.IC\_SHIELD\_MINI,  
 (20\*self.GAME.SCALE\_X, 110\*self.GAME.SCALE\_Y))  
340 self.SCREEN.blit(self.GAME.IC\_PLAYED\_TIME,  
 (15\*self.GAME.SCALE\_X, 150\*self.GAME.SCALE\_Y))  
341 self.SCREEN.blit(self.GAME.IC\_WINRATE,  
 (15\*self.GAME.SCALE\_X, 200\*self.GAME.SCALE\_Y))  
342 self.shield = View(70\*self.GAME.SCALE\_X, 125\*self.GAME.SCALE\_Y,  
 text=str(self.USER.item\_shield), gravity=**"center\_horizontal"**)  
343 self.playedTime = View(60\*self.GAME.SCALE\_X, 170\*self.GAME.SCALE\_Y,  
 text=**"time played: "**+str(self.USER.playTime), gravity=**"center\_horizontal"**)  
344 self.winrate = View(60\*self.GAME.SCALE\_X, 220\*self.GAME.SCALE\_Y,  
 text=**"win rate: "**+str(self.USER.winrate)+**"%"**, gravity=**"center\_horizontal"**)  
345 list = [self.winrate, self.playedTime, self.shield, self.playedTime,  
 self.player\_name, self.coin]  
346 **for** v **in** list:  
347 v.show()  
348 **def** is\_clicked(self):  
349 **return** pygame.mouse.get\_pressed()[0] **and** self.rect.collidepoint(pygame.mouse.get\_pos())  
350 **class** HistoryPage():  
351 **def** \_\_init\_\_(self, gameLancher):  
352 self.GAME = gameLancher  
353 self.SCREEN = pygame.display.get\_surface()  
354 self.i = 0  
355 *#add img*356 self.historyForm = self.GAME.load\_img(**"img/pg\_history\_board.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//2, self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
357 self.size = self.historyForm.get\_rect().size  
358 self.scroll = self.GAME.load\_img(**"img/ic\_scroll.png"**, self.size[0]\*7/150, -1)  
359 self.scroll\_y = self.GAME.GAME\_HEIGHT//4  
360 self.scroll\_y\_step = 0  
361 self.scroll\_x = self.GAME.GAME\_WIDTH//4 + self.size[0]\*(1 - 7/150)  
362 self.size\_scroll = self.scroll.get\_rect().size  
363 *#add text*364 self.btn\_back = View(gameLancher.GAME\_WIDTH//2, 520\*self.GAME.SCALE\_Y,  
 **"Back"**, color=**"#3ae300"**, gravity=**"center"**)  
365 self.listRacerTypeText = []  
366 self.listCoinResultText = []  
367 i = 0  
368 **while** i < 5:  
369 self.listRacerTypeText.append(gameLancher.load\_img(**"img/rc\_snail1.png"**, -1, 50))  
370 self.listCoinResultText.append(View(self.GAME.GAME\_WIDTH/1.6,  
 self.GAME.GAME\_HEIGHT/2.6 + (i + 1)\*50\*self.GAME.SCALE\_Y, color=**"#FFFFFF"**))  
371 i += 1  
372 self.typeTitle = View(self.GAME.GAME\_WIDTH/3.5, self.GAME.GAME\_HEIGHT/2.6,  
 text=**"Type"**, color=**"#FFFFFF"**)  
373 self.coinTitle = View(self.GAME.GAME\_WIDTH/1.6, self.GAME.GAME\_HEIGHT/2.6,  
 text=**"Result"**, color=**"#FFFFFF"**)  
374  
375 self.warning = View(self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT/2.6,  
 text=**"You have not played yet"**, color=**"#FFFFFF"**, gravity=**"center"**)  
376 **pass**377   
378 **def** Up(self):  
379 self.i -= 1  
380 self.scroll\_y -= self.scroll\_y\_step  
381   
382 **def** Down(self):  
383 self.i += 1  
384 self.scroll\_y += self.scroll\_y\_step  
385   
386 **def** setHistory(self, history):  
387 *#ensure self.i is valid*388 **if** (self.i < 0):  
389 self.i = 0  
390 self.scroll\_y += self.scroll\_y\_step  
391 **elif** self.i + 2 >= len(history):  
392 self.i = max(len(history) - 3, 0)  
393 self.scroll\_y -= self.scroll\_y\_step  
394 *# set len of scroll*395 len\_h = len(history) - 2  
396 **if** len\_h <= 0:  
397 len\_h = 1  
398 self.scroll = pygame.transform.scale(self.scroll,  
 (self.size\_scroll[0], int(self.size[1]/len\_h)))  
399 self.scroll\_y\_step = self.size[1]/len\_h  
400 **for** i **in** range(0, 3):  
401 **if**(i < len(history)):  
402 self.listRacerTypeText[i] = self.GAME.load\_img(**"img/"**+history[len(history)  
 - (self.i + i + 1)].racerType + **".png"**, -1, 50)  
403 *#self.listRacerNumText[i].setText(history[len(history) - i].racerNum)*404 self.listCoinResultText[i].setText(history[len(history)  
 - (self.i + i + 1)].coinResult)  
405   
406 **def** draw(self, history):  
407 self.SCREEN.blit(self.historyForm, (self.GAME.GAME\_WIDTH//4, self.GAME.GAME\_HEIGHT//4))  
408   
409 **if** (len(history) == 0):  
410 self.warning.show()  
411 self.btn\_back.show()  
412 **else**:  
413 **for** i **in** range(0,3):  
414 **if** (i < len(history)):  
415 self.SCREEN.blit(self.listRacerTypeText[i],(self.GAME.GAME\_WIDTH//3.6,  
 self.GAME.GAME\_HEIGHT//2.7 + (i + 1)\*50\*self.GAME.SCALE\_Y))  
416 self.listCoinResultText[i].show()  
417 self.coinTitle.show()  
418 self.typeTitle.show()  
419 self.btn\_back.show()  
420 *# show scroll bar*421 self.SCREEN.blit(self.scroll, (self.scroll\_x, self.scroll\_y))  
422 **pass**423 **class** Shoppage():  
424 **def** \_\_init\_\_(self, gameLancher,user):  
425 self.GAME = gameLancher  
426 self.SCREEN = pygame.display.get\_surface()  
427 self.SHOP = self.GAME.load\_img(**"img/pg\_mainpage\_no\_title.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//2,self.GAME.GAME\_HEIGHT\_DEFAULT//2)  
428 self.SHIELD = self.GAME.load\_img(**"img/ic\_shield.png"**,  
 self.GAME.GAME\_WIDTH\_DEFAULT//10, self.GAME.GAME\_HEIGHT\_DEFAULT//7)  
429 self.rect = self.SHOP.get\_rect()  
430 self.rect.center = (self.GAME.GAME\_WIDTH//2, self.GAME.GAME\_HEIGHT//2)  
431 self.TITLE = View(self.rect.x + self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.0535, text=**"SHOP"**, gravity=**"center"**)  
432 self.btn\_back = View(self.rect.x +self.rect.w\*0.5,  
 self.rect.y + self.rect.h\*0.8226, text=**"Back"**, gravity=**"center"**)  
433 self.val\_price\_shield = 1000  
434 self.tv\_shield = View(self.rect.x +self.rect.w\*0.32,  
 self.rect.y + self.rect.h\*0.279, **"SHIELD"**, gravity=**"center\_horizontal"**)  
435 self.tv\_shield\_val = View(self.rect.x +self.rect.w\*0.32, self.rect.y +  
 self.rect.h\*0.455, text=**"You own: "**+str(user.item\_shield),  
 gravity=**"center\_horizontal"**)  
436 self.price\_shield = View(self.rect.x +self.rect.w\*0.743,  
 self.rect.y + self.rect.h\*0.455, str(self.val\_price\_shield)+**"$"**)  
437   
438 self.tv\_coin = View(self.rect.x + self.rect.w\*0.084,  
 self.rect.y + self.rect.h\*0.7, **"ADD MONEY"**, gravity=**"center\_horizontal"**)  
439 self.buy\_coin = View(self.rect.x + self.rect.w\*0.743,  
 self.rect.y + self.rect.h\*0.7, **"BUY 10000$"**, gravity=**"center"**)  
440 self.list\_view = [self.TITLE, self.btn\_back, self.tv\_shield\_val,  
 self.tv\_shield, self.price\_shield, self.tv\_coin, self.buy\_coin]  
441 **pass**442 **def** DrawShop(self, user):  
443 self.TITLE.setText(**"SHOP"**)  
444 self.tv\_shield\_val.setText(**"You own: "**+str(user.item\_shield))  
445 self.SCREEN.blit(self.SHOP, self.rect)  
446 self.SCREEN.blit(self.SHIELD, (self.rect.x + self.rect.w\*0.084,  
 self.rect.y + self.rect.h\*0.2178))  
447 **if** int(user.coins) >= self.val\_price\_shield:  
448 self.price\_shield.setBackground(gameLancher=self.GAME, img\_link=**"img/S\_price1.png"**)  
449 **else**:  
450 self.price\_shield.setBackground(gameLancher=self.GAME, img\_link=**"img/S\_price.png"**)  
451 self.buy\_coin.setBackground(gameLancher=self.GAME, img\_link=**"img/S\_price1.png"**)  
452 **for** v **in** self.list\_view:  
453 v.show()  
454 **pass**455

*File WARPlayHistory.py:* đọc và ghi lịch sử

001 **import** os  
002 direct = os.getcwd()  
003 *#change direction*004 flag = **True**005 currentDirrect = **"\libs"**006 **for** i **in** range(1,len(currentDirrect)+1):  
007 a = direct[-i]  
008 b = currentDirrect[-i]  
009 **if** direct[-i] != currentDirrect[-i]:  
010 flag = **False**011 **if** flag: *# correct*012 dataDirect = **""**013 i = 0  
014 **while**(i < len(direct) - len(currentDirrect)):  
015 dataDirect = dataDirect + direct[i]  
016 i = i + 1  
017 os.chdir(dataDirect)  
018   
019 **from** global\_variables **import** \*  
020 **from** decryptData **import** \*  
021 **from** encryptdata **import** \*  
022 **else**:  
023 **from** libs.global\_variables **import** \*  
024 **from** libs.decryptData **import** \*  
025 **from** libs.encryptdata **import** \*  
026   
027 endOfData = **"\""** *#to define where to stop read data*028 racerTypeTitle = **"racerType = \""**029 racerNumTitle = **"racerNum = \""**030 coinResulTitle = **"coinResult = \""**031 key = 123  
032   
033 **class** ReadHistoryData:  
034 **def** StringCompare(a, b, startPossOfA):  
035 i = startPossOfA  
036 j = 0  
037 **while**((i < len(a)) & (j < len(b))):  
038 **if**(a[i] != b[j]):  
039 **return False**040 i = i + 1  
041 j = j + 1  
042 **return True**043   
044 **def** StringCopy(src, startPoss):  
045 **if**(startPoss < len(src) - 1):  
046 dest = src[startPoss]  
047 i = startPoss + 1  
048 **while**(src[i] != endOfData):  
049 dest = dest + src[i]  
050 i = i + 1  
051 **return** dest  
052   
053 **def** GetData(data, startPoss, history):  
054 i = startPoss  
055 *# get type*056 **if** (ReadHistoryData.StringCompare(data, racerTypeTitle, i)):  
057 i = i + len(racerTypeTitle)  
058 history.append(History())  
059 history[len(history) - 1].racerType = ReadHistoryData.StringCopy(data, i)  
060 **else**:  
061 **return** -1  
062 *#get number*063 i = i + len(history[len(history) - 1].racerType) + 2  
064 **if** (ReadHistoryData.StringCompare(data, racerNumTitle, i)):  
065 i = i + len(racerNumTitle)  
066 history[len(history) - 1].racerNum = ReadHistoryData.StringCopy(data, i)  
067 **else**:  
068 **return** -1  
069 *#get result*070 i = i + len(history[len(history) - 1].racerNum) + 2 *#change poss to pass....*071 **if** (ReadHistoryData.StringCompare(data, coinResulTitle, i)):  
072 i = i + len((coinResulTitle))  
073 history[len(history) - 1].coinResult = ReadHistoryData.StringCopy(data, i)  
074 **else**:  
075 **return** -1  
076 i = i + len(str(history[len(history) - 1].coinResult)) + 2 *#change pos to end history*077 **return** i  
078 *#ham tra ve -1 neu khong doc duoc*079 **def** GetAllHistoryData(userID, history):  
080 dataLocation = **"data/"** + str(userID) + **".txt"**081 **try**:  
082 f = open(dataLocation, **"rt"**)  
083 **except** FileNotFoundError:  
084 *# ERROR ACCESS FILE*085 **return**086   
087 data = f.read()  
088 dataLength = len(data)  
089 i = 0  
090 **while**(i < dataLength - 20):  
091 i = ReadHistoryData.GetData(data, i, history) + 1  
092 **if**(i == -1): *# HISTORY ERROR*093 **return** -1  
094 **return** history  
095 **pass**096   
097 **class** WriteHistoryData:  
098 **def** WriteAllHistoryData(userID, history):  
099 dataLocation = **"data/"** + str(userID) + **".txt"**100 f = open(dataLocation, **"w"**)  
101 i = 0  
102 textToWrite = **""**103 **while**(i < len(history)):  
104 textToWrite += racerTypeTitle + history[i].racerType + **"\" "**105 textToWrite += racerNumTitle + history[i].racerNum + **"\" "**106 textToWrite += coinResulTitle + history[i].coinResult + **"\" "**107 textToWrite = textToWrite + **'\n'**108 i = i + 1  
109 f.write(textToWrite)  
110 **pass**111 **'''  
112 history = []  
113 ReadHistoryData.GetAllHistoryData("100000", history)  
114 '''**

*File WARUserData.py:* Đọc và ghi dữ liệu

001 **import** os  
002 direct = os.getcwd()  
003 *#change direction*004   
005 flag = **True**006 currentDirrect = **"\libs"**007 **for** i **in** range(1,len(currentDirrect)+1):  
008 a = direct[-i]  
009 b = currentDirrect[-i]  
010 **if** direct[-i] != currentDirrect[-i]:  
011 flag = **False**012 **if** flag:  
013 dataDirect = **""**014 i = 0  
015 **while**(i < len(direct) - len(currentDirrect)):  
016 dataDirect = dataDirect + direct[i]  
017 i = i + 1  
018 os.chdir(dataDirect)  
019   
020 **from** global\_variables **import** \*  
021 **from** decryptData **import** \*  
022 **from** encryptdata **import** \*  
023 **else**:  
024 **from** libs.global\_variables **import** \*  
025 **from** libs.decryptData **import** \*  
026 **from** libs.encryptdata **import** \*  
027   
028 *#key to encript and decrip from 1 to 127*029 key = 125  
030 dataLocation = **"data/usersData.txt"**031 userIDTitle = **"ID = \""**032 nameTitle = **"username = \""**033 passTitle = **"password = \""**034 coinsTitle = **"coins = \""**035 playTimeTitle = **"playTime = \""**036 winrateTitle = **"winrate = \""**037 shieldTitle = **"shield = \""**038 endOfData = **"\""** *#to define where to stop read data*039   
040 **class** ReadUsersData:  
041   
042 **def** StringCompare(a, b, startPossOfA):  
043 i = startPossOfA  
044 j = 0  
045 **while**((i < len(a)) & (j < len(b))):  
046 **if**(a[i] != b[j]):  
047 **return False**048 i = i + 1  
049 j = j + 1  
050 **return True**051   
052 **def** StringCopy(src, startPoss):  
053 **if**(startPoss < len(src) - 1):  
054 dest = src[startPoss]  
055 i = startPoss + 1  
056 **while**((src[i] != endOfData)):  
057 dest = dest + src[i]  
058 i = i + 1  
059 **if**(i == len(src)):  
060 **break**;  
061 **return** dest  
062   
063 **def** GetUserData(data, startPoss, user):  
064 i = startPoss  
065 *# get ID*066 **if** (ReadUsersData.StringCompare(data, userIDTitle, i)):  
067 i = i + len(userIDTitle)  
068 user.append(User())  
069 ID = ReadUsersData.StringCopy(data, i)  
070 **if** (ID == **""**):  
071 **return** -1  
072 user[len(user) - 1].ID = int(ID)  
073 **else**:  
074 **return** -1  
075 *#get name*076 i = i + len(ID) + 2  
077 **if** (ReadUsersData.StringCompare(data, nameTitle, i)):  
078 i = i + len(nameTitle)  
079 user[len(user) - 1].name = ReadUsersData.StringCopy(data, i)  
080 **if**(user[len(user) - 1].name == **""**):  
081 **return** -1  
082 **else**:  
083 **return** -1  
084 *#get pass*085 i = i + len(user[len(user) - 1].name) + 2 *#change poss to pass....*086 **if** (ReadUsersData.StringCompare(data, passTitle, i)):  
087 i = i + len((passTitle))  
088 user[len(user) - 1].password = ReadUsersData.StringCopy(data, i)  
089 **if** (user[len(user) - 1].password == **""**):  
090 **return** -1  
091 **else**:  
092 **return** -1  
093 *#get coins*094 i = i + len(user[len(user) - 1].password) + 2 *#change poss to coins....*095 **if** (ReadUsersData.StringCompare(data, coinsTitle, i)):  
096 i = i + len((coinsTitle))  
097 user[len(user) - 1].coins = ReadUsersData.StringCopy(data, i)  
098 **if** (user[len(user) - 1].coins == **""**):  
099 **return** -1  
100 **else**:  
101 **return** -1  
102 *#get playtime*103 i = i + len(str(user[len(user) - 1].coins)) + 2 *#change poss to playtime....*104 **if** (ReadUsersData.StringCompare(data, playTimeTitle, i)):  
105 i = i + len((playTimeTitle))  
106 user[len(user) - 1].playTime = ReadUsersData.StringCopy(data, i)  
107 **if** (user[len(user) - 1].playTime == **""**):  
108 **return** -1  
109 **else**:  
110 **return** -1  
111 *# get winrate*112 i = i + len(str(user[len(user) - 1].playTime)) + 2 *# change poss to winrate....*113 **if** (ReadUsersData.StringCompare(data, winrateTitle, i)):  
114 i = i + len((winrateTitle))  
115 user[len(user) - 1].winrate = ReadUsersData.StringCopy(data, i)  
116 **if** (user[len(user) - 1].winrate == **""**):  
117 **return** -1  
118 **else**:  
119 **return** -1  
120 *# get shell number*121 i = i + len(str(user[len(user) - 1].winrate)) + 2 *# change poss to shell....*122 **if** (ReadUsersData.StringCompare(data, shieldTitle, i)):  
123 i = i + len((shieldTitle))  
124 shield = ReadUsersData.StringCopy(data, i)  
125 **if**(shield == **None**):  
126 **return** -1  
127 user[len(user) - 1].item\_shield = int(shield)  
128 **else**:  
129 **return** -1  
130 i = i + len(shield) + 2 *#change poss to end of user*131 **return** i  
132   
133 *#ham tra ve -1 neu khong doc duoc*134 **def** GetAllUsersData(user):  
135 **try**:  
136 f = open(dataLocation, **"rt"**)  
137 **except** FileNotFoundError:  
138 **return** -1  
139 data = f.read()  
140 data = decrypt.DecyptData(data, key)  
141 dataLength = len(data)  
142 i = 0  
143 **while**(i < dataLength - 20):  
144 i = ReadUsersData.GetUserData(data, i, user)  
145 **if**(i == -1):  
146 *# ERROR*147 **return** -2  
148 **pass**149   
150 **class** WriteUsersData:  
151   
152 **def** WriteAllUsersData(user):  
153 f = open(dataLocation, **"w"**)  
154 i = 0  
155 textToWrite = **""**156 **while**(i < len(user)):  
157 textToWrite += userIDTitle + str(user[i].ID) + **"\" "**158 textToWrite += nameTitle + user[i].name + **"\" "**159 textToWrite += passTitle + user[i].password + **"\" "**160 textToWrite += coinsTitle + user[i].coins + **"\" "**161 textToWrite += playTimeTitle + user[i].playTime + **"\" "**162 textToWrite += winrateTitle + user[i].winrate + **"\" "**163 textToWrite += shieldTitle + str(user[i].item\_shield) + **"\" "**164 i = i + 1  
165 textToWrite = encypt.EncryptData(textToWrite, key)  
166 f.write(textToWrite)  
167 **pass**168   
169 **class** LoginCore:  
170 **def** FindUserName(listUser, x):  
 *#return -1 if not exist -2 if wrong pass or poss of user in list*171 **if**(len(listUser) == 0):  
172 **return** -1  
173 i = 0  
174 exist = **False**175 rightPass = **False**176 poss = 0  
177 **while** i < len(listUser):  
178 **if**((ReadUsersData.StringCompare(listUser[i].name, x.name, 0))  
 & (len(listUser[i].name) == len(x.name))):  
179 exist = **True**180 poss = i  
181 **break**182 i += 1  
183 **if not** exist:  
184 **return** -1  
185   
186 **if** ((ReadUsersData.StringCompare(listUser[i].password, x.password, 0))  
 & (len(listUser[i].password) == len(x.password))):  
187 rightPass = **True**188 **if not** rightPass:  
189 **return** -2  
190 **return** poss  
191   
192 **def** CheckName(user):  
193 i = 0  
194 **while** i < len(user.name):  
195 **if**(user.name[i] == endOfData):  
196 **return False**197 i += 1  
198 i = 0  
199 **while** i < len(user.password):  
200 **if**(user.password[i] == endOfData):  
201 **return False**202 i += 1  
203 **return True**204 **pass**

*File Widgets.py*: Chứa các tiện ích mở rộng

001 **from** colormap.colors **import** hex2rgb  
002 **from** libs.global\_variables **import** \*  
003 **import** pygame  
004   
005 **class** View(pygame.sprite.Sprite):  
006 **def** \_\_init\_\_(self, startX, startY, text=**"View"**, color=**"#B33333"**,  
 bgrColor=**"#FFFFFF"**, gravity=**"center"**) -> **None**:  
007 super().\_\_init\_\_()  
008 self.startX = startX  
009 self.startY = startY  
010 self.text = text  
011 self.bgrColor = bgrColor  
012 self.rect = pygame.Rect(self.startX, self.startY, 0, 0)  
013 self.rect.normalize()  
014 self.surface = **None**015 self.bk\_surf = self.surface  
016 self.isTransparent = **True**017 self.color = color  
018 self.gravity = **"top\_left"**019 self.setGravity(gravity)  
020 self.SCREEN = pygame.display.get\_surface()  
021 self.bgrImg = **None**022 **def** setGravity(self, gravity):  
023 list\_gravity = [**"top\_left"**, **"bottom\_left"**, **"center"**, **"center\_vertical"**,  
 **"center\_horizontal"**, **"top\_right"**, **"bottom\_right"**,  
024 **"mid\_bottom"**, **"mid\_left"**, **"mid\_right"**]  
025 **if** gravity **in** list\_gravity:  
026 self.gravity = gravity  
027 **def** show(self):  
028 **if** self.bgrImg != **None**:  
029 self.SCREEN.blit(self.bgrImg, (self.rect.x - 10, self.rect.y - 5))  
030 self.rect = showText(self.startX, self.startY, self.text,  
 color=self.color, isTransparent=self.isTransparent, gravity=self.gravity)  
031 **def** setText(self, text):  
032 self.text = text  
033 self.show()  
034 **def** setBackground(self, gameLancher, img\_link=**None**):  
035 **if** img\_link != **None**:  
036 self.bgrImg = gameLancher.load\_img(img\_link, self.rect.w + 20, self.rect.h + 10)  
037 **else**:  
038 self.bgrImg = **None**039 self.show()  
040 **def** is\_clicked(self, gameLancher):  
041 **if** self.rect.collidepoint(pygame.mouse.get\_pos()):  
042 self.setBackground(gameLancher,**"img/bg\_button\_isclicked.png"**)  
043 **else**:  
044 self.setBackground(gameLancher=gameLancher)  
045 **return** pygame.mouse.get\_pressed()[0] **and** self.rect.collidepoint(pygame.mouse.get\_pos())  
046 **pass**047 COLOR\_INACTIVE = pygame.Color(**'lightskyblue3'**)  
048 COLOR\_ACTIVE = pygame.Color(**'dodgerblue2'**)  
049 **class** InputBox:  
050 **def** \_\_init\_\_(self, x, y, w, h, text=**''**, isdigit=**False**, ispassword =**False**):  
051 self.rect = pygame.Rect(x, y, w, h)  
052 self.color = COLOR\_INACTIVE  
053 self.text = text  
054 self.FONT = pygame.font.SysFont(**'Comic Sans MS'**, 30)  
055 self.txt\_surface = self.FONT.render(text, **True**, self.color)  
056 self.active = **False**057 self.hidetext = text  
058 self.isPassword = ispassword  
059 self.isDigit = isdigit  
060   
061 **def** handle\_event(self, event):  
062 **if** event.type == pygame.MOUSEBUTTONDOWN:  
063 self.active = self.rect.collidepoint(event.pos)  
064 *# Change the current color of the input box.*065 self.color = COLOR\_ACTIVE **if** self.active **else** COLOR\_INACTIVE  
066 **if** event.type == pygame.KEYDOWN:  
067 **if** self.active:  
068 **if** event.key == pygame.K\_BACKSPACE:  
069 self.text = self.text[:-1]  
070 **if** self.isPassword:  
071 self.hidetext = self.hidetext[:-1]  
072 self.txt\_surface = self.FONT.render(self.text, **True**, self.color)  
073 **return** 1  
074 **else**:  
075 **if** len(self.text) >= 10:  
076 **return** 0  
077 buffer = **""** + str(event.unicode)  
078 **if** self.isDigit **and not** buffer.isdigit():  
079 buffer = **""**080 **if** self.isPassword:  
081 self.hidetext += buffer  
082 **if** len(self.hidetext) > len(self.text):  
083 self.text += **"\*"**084 **else**:  
085 self.text += buffer  
086 self.txt\_surface = self.FONT.render(self.text, **True**, self.color)  
087 **return** 1  
088 self.txt\_surface = self.FONT.render(self.text, **True**, self.color)  
089 **return** -1  
090 **def** draw(self, screen):  
091 screen.blit(self.txt\_surface, (self.rect.x+5, self.rect.y+5))  
092 pygame.draw.rect(screen, self.color, self.rect, 2)  
093   
094 **class** ImageView():  
095 **def** \_\_init\_\_(self, game, x, y, w, h, img\_link):  
096 self.x = x  
097 self.y = y  
098 self.color = COLOR\_INACTIVE  
099 self.game = game  
100 self.IMAGE = self.game.load\_img(img\_link, w,h)  
101 self.rect = self.IMAGE.get\_rect()  
102 self.rect.x = self.x  
103 self.rect.y = self.y  
104 self.active = **False**105 **def** setActive(self,isActive=**False**):  
106 self.active = isActive  
107 self.color = COLOR\_ACTIVE **if** self.active **else** COLOR\_INACTIVE  
108 **def** is\_clicked(self):  
109 **return** pygame.mouse.get\_pressed()[0] **and** self.rect.collidepoint(pygame.mouse.get\_pos())  
110 **def** draw(self, screen):  
111 screen.blit(self.IMAGE, (self.x, self.y))  
112 pygame.draw.rect(screen, self.color, self.rect, 2)  
113 **pass**114   
115 **def** showText( x, y, text=**"View"**, font=**"freesansbold.ttf"**,  
 color=**"#FFFFFF"**, textSize=20, bgrColor=**None**,isTransparent=**True**, gravity=**"top\_left"**):  
116 font = pygame.font.SysFont(**'Comic Sans MS'**, textSize)  
117 **if** isTransparent:  
118 text = font.render(text, **True**, hex2rgb(color, normalise=**False**))  
119 **elif** bgrColor!=**None**:  
120 text = font.render(text, **True**, hex2rgb(color, normalise=**False**),  
 hex2rgb(bgrColor, normalise=**False**))  
121 **else**:  
122 text = font.render(text, **True**, hex2rgb(color, normalise=**False**), bgrColor)  
123 textRect = text.get\_rect()  
124 **'''  
125 $ checkout the gravity  
126 The Rect object has several virtual attributes which can be used to move and**

**align the Rect:  
127 x,y  
128 top, left, bottom, right  
129 topleft, bottomleft, topright, bottomright  
130 midtop, midleft, midbottom, midright  
131 center, centerx, centery  
132 size, width, height  
133 w,h  
134 '''**135 list\_gravity = [**"top\_left"**, **"bottom\_left"**, **"center"**, **"center\_vertical"**,  
 **"center\_horizontal"**, **"top\_right"**,  
136 **"bottom\_right"**, **"mid\_bottom"**, **"mid\_left"**, **"mid\_right"**]  
137 **if** gravity == **"top\_left"**:  
138 textRect.topleft = (x, y)  
139 **elif** gravity == **"bottom\_left"**:  
140 textRect.bottomleft =(x,y)  
141 **elif** gravity ==**"center"**:  
142 textRect.center = (x,y)  
143 **elif** gravity ==**"center\_vertical"**:  
144 textRect.centerx = x  
145 textRect.y = y  
146 **elif** gravity ==**"center\_horizontal"**:  
147 textRect.centery = y  
148 textRect.x = x  
149 **elif** gravity ==**"top\_right"**:  
150 textRect.topright = (x,y)  
151 **elif** gravity == **"bottom\_right"**:  
152 textRect.bottomright = (x, y)  
153 **elif** gravity == **"mid\_bottom"**:  
154 textRect.midbottom = (x,y)  
155 **elif** gravity == **"mid\_left"**:  
156 textRect.midleft = (x, y)  
157 **elif** gravity == **"mid\_right"**:  
158 textRect.midright = (x, y)  
159 **else**:  
160 *# set default if error*161 textRect.topleft = (x, y)  
162   
163 screen = pygame.display.get\_surface()  
164 screen.blit(text, textRect)  
165 **return** textRect

166

|  |
| --- |
| **PHỤ LỤC** |

Các tài liệu kèm:

* Project Plan
* Biên bản họp nhóm các tuần
* Biên bản cam kết
* Tài liệu hướng dẫn sử dụng sản phẩm

|  |
| --- |
| **TÀI LIỆU THAM KHẢO** |

[3]: References of hex2rgb

<https://pythonhosted.org/colormap/references.html#colormap.colors.hex2rgb>

[3]:

Pygame