Wisielec

1. Wizja projektu

Koncepcja projektu

Gra polega na odgadnięciu losowo wybranego słowa z dowolnej kategorii, litera po literze. Każda litera występująca w haśle przybliża gracza do zwycięstwa. Każda nie występująca w haśle litera powoduje dorysowywanie kolejnego elementu wisielca, po narysowaniu całego rysunku, przegrywasz! Aplikacja umożliwia też możliwość gry na czas, dla tych którzy lubią większe wyzwania. Do bazy danych można też dodawać swoje słowa, a ranking najlepszych rozgrywek jest zapisywany przez aplikacje.

Wymagania

Narzucone przez drugi zespół:
1. Ranking - Możliwość sprawdzenia swojego dotychczasowego wyniku we wszystkich grach, tak jak np. w pasjansie na windowsie XP. 5 najlepszych wyników wszechczasów wczytanych z serializowanego pliku zostaje na tablicy wyników.
2. Widoczny szary wisielec - cały wisielec jest widoczny od początku, ale jest w bardziej szarym odcieniu. Pomaga graczowi zdać sobie sprawę, ile jeszcze "żyć" jest do przegranej.
3. Serializacja (zapisanie stanu gry) - W każdym momencie można przerwać grę, i rozpocząć od tego samego momentu. Niedostępne w trybie gry na czas. Dodatkowo, serializowane są najlepsze wyniki po każdej wygranej rozgrywce.
4. Tryb nocny- tryb z ciemnym tłem, który nie jest po prostu negacją. Nie razi w oczy podczas używania w nocy.
5. Wisielec robi pajacyka po kliknięciu na niego - nie trzeba za dużo tłumaczyć.
6. Dźwięki przy poprawnej/niepoprawnej literze - ewentualnie jakieś dodatkowe dźwięki w wypadku przegranej, wygranej.

Jedno z wymagań od drugiego zespołu tylko lekko zmodyfikowaliśmy:
7. 3 poziomy trudności - W zależności od długości słowa. Determinuje też ilość punktów w czasie rozgrywki.

Na koniec wymagania dodane przez nas:
8. Tryb gry na czas - dla większego wyzwania, zegar jest widoczny w czasie gry, jeśli wybierzemy taki tryb gry.

9. Możliwość dodawania własnych słów do bazy danych.
10. Kategorie, które podpowiadają o czym jest hasło - można wybrać kategorię, z których będą losowane hasła do gry. Każda kategoria będzie miała bazę danych ze słowami, z których hasła będą losowane do każdej gry. Co najmniej 7, każda z co najmniej 3 słowami na poziom trudności.

2. Opis funkcjonalności

Zgodnie z koncepcją projektu, musieliśmy stworzyć grę zgodną z założeniami wisielca. W celu stworzenia graficznego interfejsu, skorzystaliśmy z JavaFX i programu SceneBuilder. Korzystaliśmy z wzorca projektowego Model-View-Controller. Logika aplikacji była oddzielona od tego co widział użytkownik, a komunikacja pomiędzy nimi zachodzi dzięki klasom-kontrolerom.
Planowanie zaczęliśmy od ekranu gry. Założyliśmy, że słowo do odgadnięcia należy przedstawić w postaci odpowiedniej ilości znaków “_”, z widocznymi odstępami między nimi. Zgadywanie liter odbywa się poprzez naciśnięcie odpowiednich klawiszy, każdy odpowiadający innej literze alfabetu polskiego lub angielskiego. Na tym samym ekranie musiał być też widoczny wisielec, oraz przyciski do zapisu gry i poddania się. Dodatkowo dodaliśmy możliwość wpisania i sprawdzenia całego hasła, co powoduje automatyczną utratę wszystkich żyć w przypadku niepowodzenia. Następnie powstał widok odpowiadający za menu główne, dodawanie hasła, kolejne widoki na których gracze mogą wybierać tryby gry, oraz widok rankingu.
Równolegle z widokami, powstawały kolejne funkcjonalności związane z logiką gry i aplikacji. Przechodzenie pomiędzy widokami, poprawne wyświetlanie haseł i liter, a także wczytywanie danych oraz . W celu zapisywania danych, skorzystaliśmy z serializacji z wbudowanych bibliotek Javy, m. in. interfejsu Serializable.

3. Diagramy klas

Diagramy klas znajdują się w folderze Class Diagrams, dostępnym pod linkiem:
https://github.com/FearlessPoro/IO-Wisielec/tree/WIP/DOC/Class%20diagrams
Zostały one wygenerowane przy użyciu wtyczki Intellij.
Do trzech funkcjonalności: uruchamiania trybu nocnego, wyświetlania rankingu oraz wyboru parametrów rozgrywki, stworzyliśmy diagramy sekwencji i diagramy aktywności, dostępne również na naszym gicie.
Diagramy sekwencji:
https://github.com/FearlessPoro/IO-Wisielec/tree/WIP/DOC/Sequence%20Diagrams
Diagramy aktywności:
https://github.com/FearlessPoro/IO-Wisielec/tree/WIP/DOC/Activity%20diagrams
Diagram przypadków użycia, znajduje się pod adresem:

https://github.com/FearlessPoro/IO-Wisielec/blob/WIP/DOC/Diagram%20Przypadkow%20Uzycia.jpg

4. Opis zastosowanych rozwiązań

Do konfiguracji projektu korzystaliśmy z Gradle.
W celu stworzenia graficznego interfejsu użytkownika, korzystaliśmy z JavaFX. Pliki .fxml zostały wygenerowane korzystając z programu SceneBuilder.
W czasie projektu korzystaliśmy z metodologii agile, z pomocą serwisu trello, na który trafiały kolejne taski do wykonania przez członków zespołu.
Do integracji ciągłej korzystaliśmy z TravisCI, co umożliwiało nam sprawdzenie, czy testy przechodzą po dodawaniu nowego kodu.
Dokumentacja kodu nie została stworzona, uznaliśmy że kod jest samodokumentujący się.

W skład projektu wchodzi folder z kodem, src. W jego wnętrzu znajdują się dwa foldery, main oraz tests. W folderze main znajduje się kod źródłowy oraz surowce:
Kod źródłowy został podzielony na 4 paczki znajdujące się w folderze java: logic, entity, dao oraz controllers.

- Paczka logic zawiera klasy odpowiadające za logikę gry, na przykład zamianę wylosowanego słowa na string znaków “_”.
- Paczka entity zawiera klasy przechowywujące informacje, które muszą zostać zapisane w celu przywrócenia gry - status odkrytego słowa, liczbę żyć oraz to jakie litery zostały już odgadnięte.
- Paczka dao zawiera klasę, odpowiadającą za pobieranie informacji z bazy danych na temat słów losowanych do gry oraz dodawanie do tej bazy słów.
- Paczka controllers zawiera klasy odpowiadające za obsługę wydarzeń związanych z GUI. Na przykład zmianę widoków lub obsługę naciśnięć przycisków.

W folderze resources, również w folderze main znajdują się różnego rodzaju zasoby, wykorzystywane przez program.

- Folder database przechowuje baze danych słów w postaci pliku .csv.
- Folder fxml przechowuje pliki .fxml które determinują wygląd aplikacji.
- Folder images przechowuje grafiki wykorzystywane przez aplikacje.
- Folder sounds przechowuje dźwięki wykorzystywane przez aplikacje.
- Folder stylesheets przechowuje pliki .css określające style nocny i dzienny.

W folderze tests znajdują się wszystkie testy jednostkowe, dokładniej opisane w pliku Raport_testy.pdf, znajdującym się w folderze DOC.

5. Instrukcję kompilacji i obsługi.

Wszyskie informacje znajdują się w Readme projektu:
https://github.com/FearlessPoro/IO-Wisielec

6. Opis znanych błędów

Dodawanie do bazy danych słów z polskimi znakami - jest to problem polegający na odczytywaniu wpisanych danych z pola tekstowego. Pola te nie obsługują polskich znaków, przez co wpisywanie takowych do pliku w którym przechowywane są hasła kończy się wpisaniem niepoprawnego słowa, którego nie ma możliwości odgadnąć. Problemu nie da się naprawić, gdyż jest on związany z zapisywaniem danych przez TextArea.

7. Podział pracy

W plikach dokumentacji znajduje się plik programu GanttProject o nazwie IO_gantt, w którym dokładnie opisany jest podział obowiązków pomiędzy programistów oraz testerów, który został założony na początku semestru. Korzystaliśmy z metodologii Agile, wykorzystanie surowców zostało więc dodane do projektu dopiero po jego zakończeniu. Praktyka pokazała jednak, że nie wszystko da się zaplanować. Niektóre rzeczy zostały przełożone na okres wcześniejszy lub późniejszy, na przykład:

- Pajacyk został wykonany w trzecim, a nie czwartym sprincie, na prośbę Justyny Gacek.
- Zegar został wykonany po czwartym sprincie, ze względu na nieprzewidziane okoliczności, przez które jeden z programistów, Adam Andrulewicz, nie był w stanie zrobić tego na czas. Zegar został ostatecznie zaimplementowany przez Wojciecha Franczuka, w tydzień przed oddaniem projektu, równolegle z testami.
- Jeden z testerów, Piotr Pasternak nie był w stanie ukończyć swojej części unit testów na czas. Pod sam koniec projektu, kiedy testerzy zabierali się do pracy, Piotr zniknął i kontakt z nim urwał się na tydzień. Po powrocie przygotował tylko dokumentacje testów jednostkowych, pomimo licznych upomnień kierownika zespołu. Miłosz Świerad, drugi z testerów, musiał sam dokończyć pisanie testów jednostkowych oraz systemowych.

Wszystkie commity do projektu można przejrzeć na repozytorium projektu:
https://github.com/FearlessPoro/IO-Wisielec/commits/master
W zakładce Insights znajduje się też wiele ciekawych statystyk dotyczących projektu. Na przykład najwięcej commitów było w środę (wtedy kiedy laboratoria i końce sprintów).