11.05.2021 r.

Dokumentacja projektu:
Gra Tamagotchi (opieka nad zwierzątkiem)

Autor:

Michał Foks

1. Temat programu.

W grze Tamagotchi głównym zadaniem gracza było utrzymanie zwierzątka przy życiu poświęcając mu odpowiednią ilość uwagi oraz zajmując się nim. W tym projekcie zamysł gry pozostał ten sam, jednak zmodyfikowano kilka funkcjonalności oraz kilka ulepszono. Użytkownik po stworzeniu własnego konta oraz wybraniu zwierzątka, którym ma ochotę zagrać, ma możliwość karmienia, usypiania, bawienia się ze zwierzątkiem oraz dbania o jego zdrowie dzięki wizytom u lekarza (jeśli zwierzątko zachoruje). Po wybraniu danej aktywności uruchamiane są animacje.

2. Analiza tematu.

Użytkownik tworząc nowe konto, na którym może znajdować się tylko jedno zwierzę, ma do wyboru postać kota oraz psa. Po wyborze/zalogowaniu wyświetla się odpowiedni awatar oraz kilka przycisków, dzięki którym użytkownik może wywoływać akcje na swoim zwierzaku. Zadaniem gracza jest obserwować dwa wskaźniki stanu zwierzątka – zdrowia oraz samopoczucia i na ich podstawie podejmować decyzje podczas opieki. Gdy wskaźnik zdrowia osiągnie wartość 0 gra kończy się, a użytkownik ma możliwość stworzenia nowej postaci lub konta. Postęp gracza liczony jest w dniach i wyświetlany na ekranie.

Podczas tworzenia postaci losowane są charakterystyczne dla niego cechy, które wpływają na to jak zwierze reaguje na zabawy (o ile zmieniają się wskaźniki). Wskaźniki wraz z upływem czasu maleją. Każdego dnia należy na zwierzęciu wykonać każdą z trzech aktywności jedzenie, zabawa, sen. Gracz ma możliwość personalizowania menu posiłków zwierzęcia.

Podczas tworzenia projektu korzystano z programu **Qt Creator**, który jest środowiskiem programistycznym świetnie przystosowanym do prostych aplikacji okienkowych i pozwala na tworzenie różnych widoków oraz elementów programu przydatnych do komunikacji z użytkownikiem.

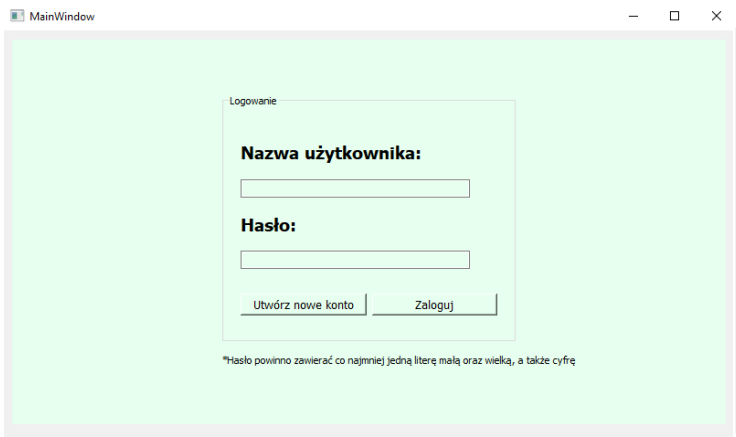
Klasą odpowiadającą za obsługę interfejsu oraz akcji i metod, które mają być wywoływane podczas wybierania konkretnych przycisków, jest klasa mainwindow, posiada ona wszelkie sloty i sygnały z elementów znajdujących się w oknie programu. Aby zapewnić odpowiednią obsługę danego użytkownika, należało stworzyć osobną klasę odpowiedzialną za czytanie i zapisywanie informacji do plików z postępem gracza oraz stworzonymi przez niego listami posiłków. Najważniejszą klasą jest klasa zwierze, w której zawierają się wszystkie informacje o zwierzęciu użytkownika podczas trwania programu oraz metody pozwalające opiekować się postacią. Jest to klasa bazowa dla klas pies oraz kot. Dzięki polimorfizmowi dodawanie kolejnych zwierząt jest bardzo proste, a klasa konkretnego zwierzęcia jest odpowiedzialna za wyświetlanie animacji. Bardzo ważną klasą jest klasa parametry, która odpowiada za ciągłe monitorowanie wskaźników zwierzęcia – zwiększanie podczas aktywności oraz zmniejszanie, gdy mijają kolejne godziny.

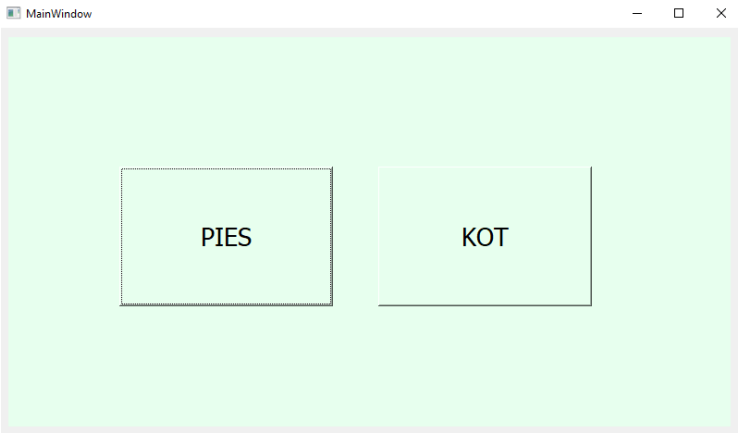
Klasa zwierze posiada obiekt klasy parametry, a także obiekt klasy choroba, który opisuje chorobę którą może zwierzątko posiadać. Aby pozwolić graczowi pomóc swojemu zwierzęciu utworzono klasę doktor, której obiekty (lekarze) pomagają w uleczeniu chorób. Równie istotnymi klasami są dziedziczące z klasy aktywności (która zawiera podstawowe informacje o aktywnościach) klasy takie jak zabawy, sen, jedzenie. Obiekty klas zabawy oraz jedzenie tworzą listy pozwalające użytkownikowi wybierać w jaki sposób zajmować się zwierzęciem. Klasa scena pozwala umieszczać obrazki na głównym ekranie.

Aby zapewnić poprawne działanie elementów dostępnych w programie Qt Creator konieczne było dołączenie bibliotek z przedrostkami Q takich jak QString, QWidget czy QListWidget. Do korzystania z możliwości języka C++ skorzystano z bibliotek takich jak memory, thread, list czy regex. Dla zapewnienia bezpieczeństwa przechowywanych haseł użytkowników skorzystano z kryptograficznego algorytmu SHA-256 i wykorzystano kod źródłowy na darmowej licencji (umieszczonej w plikach).

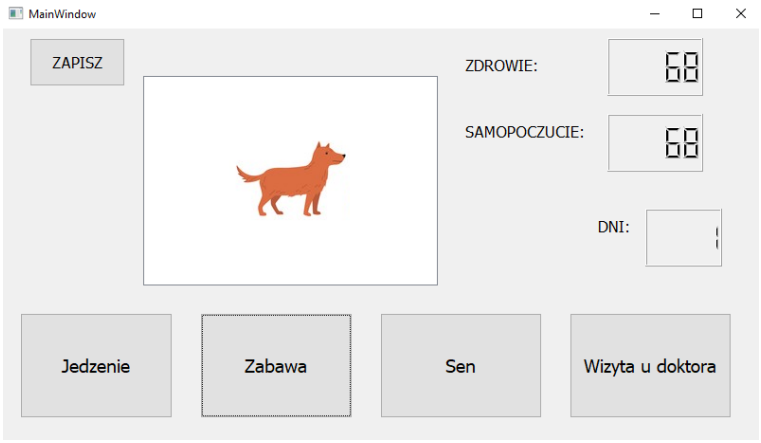
3. Specyfikacja zewnętrzna (instrukcja)

Po uruchomieniu programu należy zalogować się na konto lub utworzyć nowe i wybrać postać.



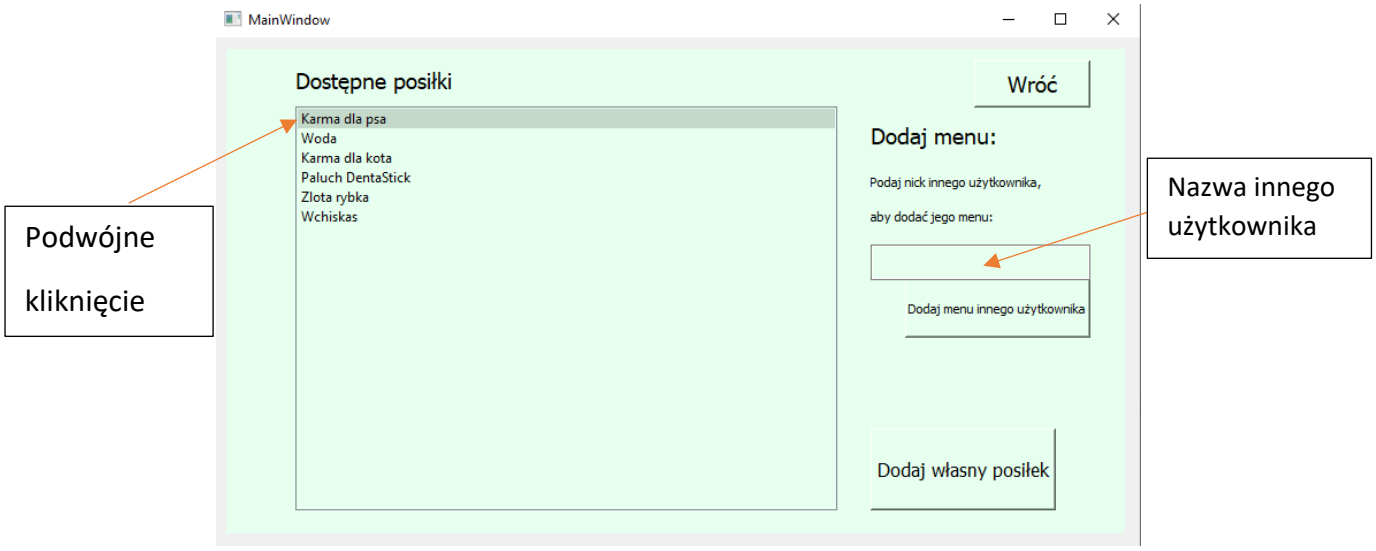


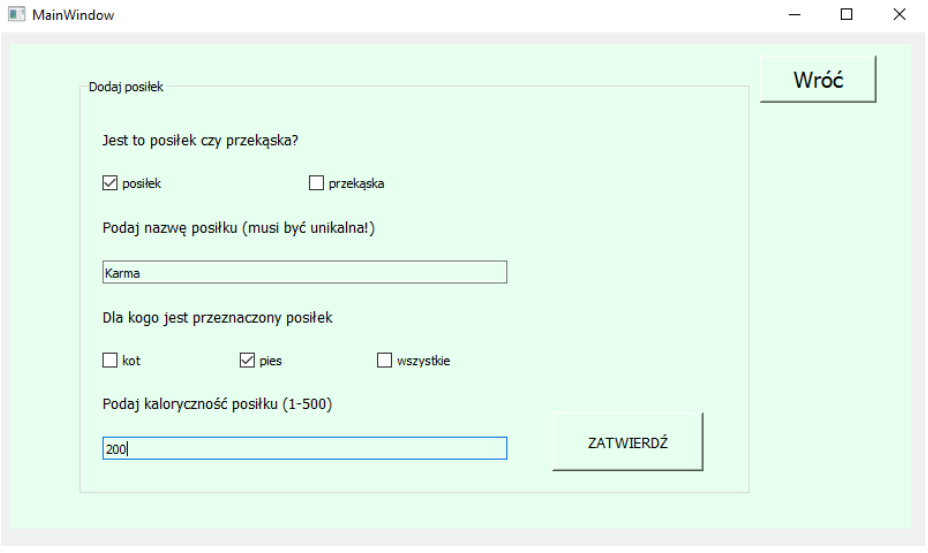
Główny ekran aplikacji wygląda następująco i pozwala poprzez naciskanie przycisków wykonywać przypisane im akcje. Poprawnie zakończyć grę można przyciskiem „Zapisz”.



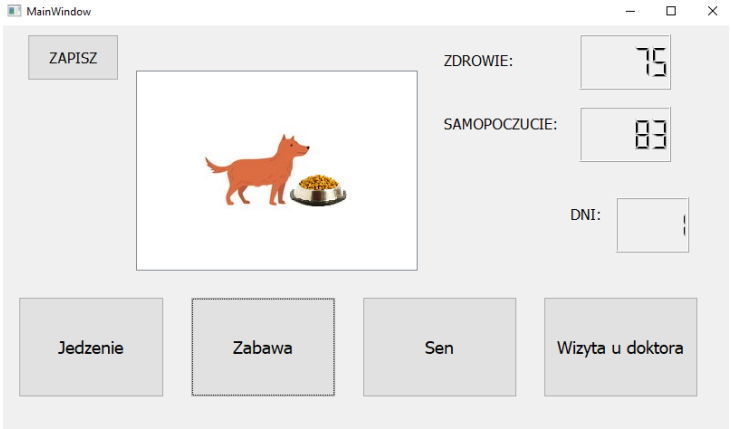
W okienku w zależności od wybranego zwierzęcia będą pojawiały się animacje.

Jeśli chcemy nakarmić nasze zwierzę należy nacisnąć przycisk „Jedzenie”, wtedy zobaczymy poniższy widok. Aby wybrać interesujący nas rekord i nakarmić zwierzę należy podwójnym kliknięciem wybrać nazwę z listy. Znając nazwę użytkownika (z naszego starego konta lub innego gracza) można zaimportować utworzony przez niego zestaw posiłków wpisując w pole po prawej stronie jego nazwę użytkownika. Możemy także dodać nasz posiłek uzupełniając formularz widoczny poniżej, następnie potwierdzając przyciskiem „Zatwierdź”.

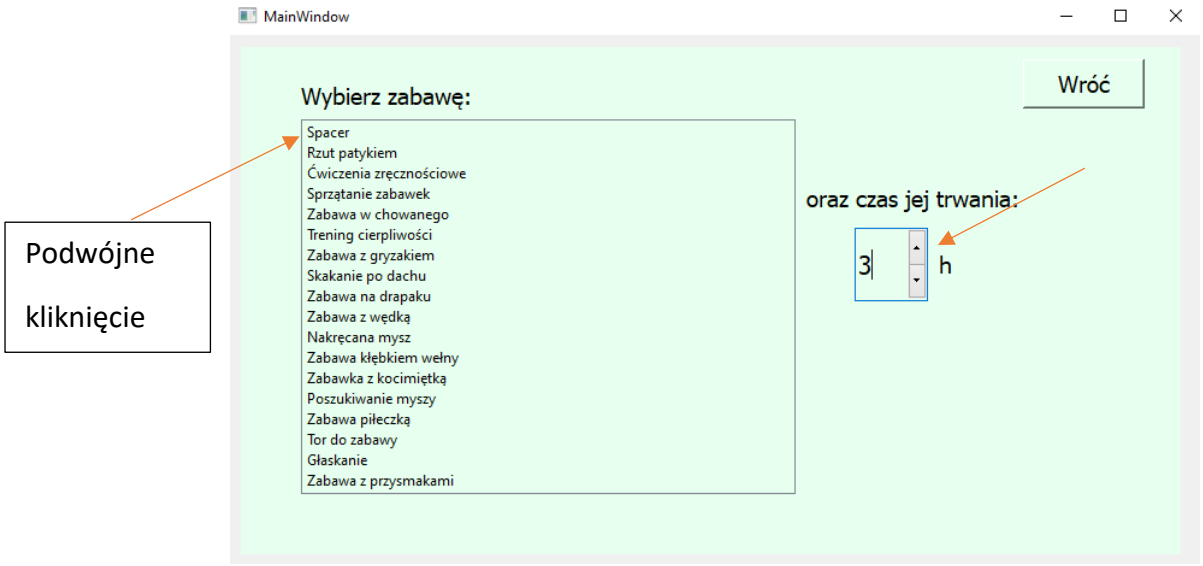




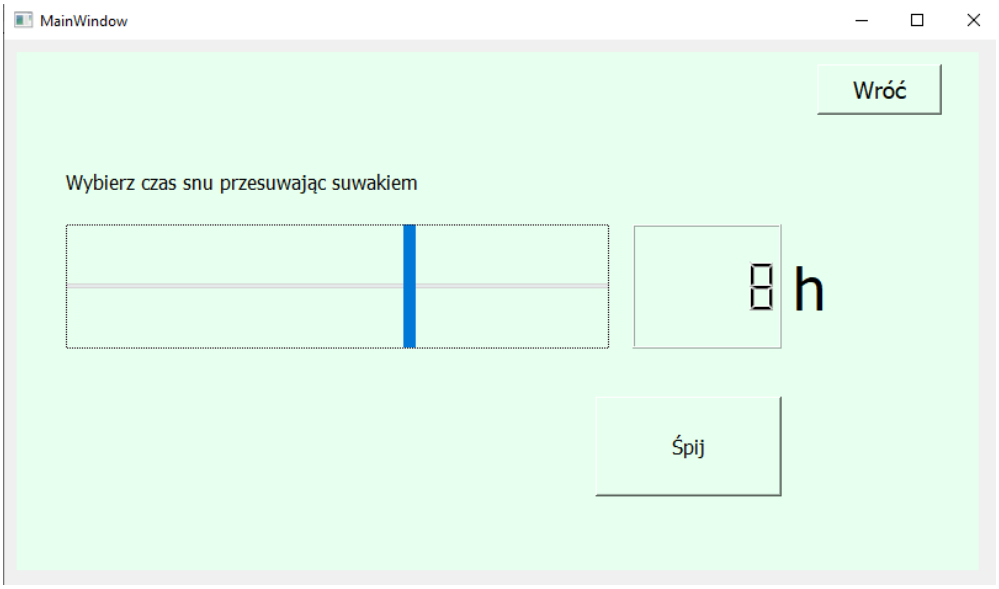
Po wybraniu posiłku pojawia się krótka animacja w głównym widoku aplikacji.



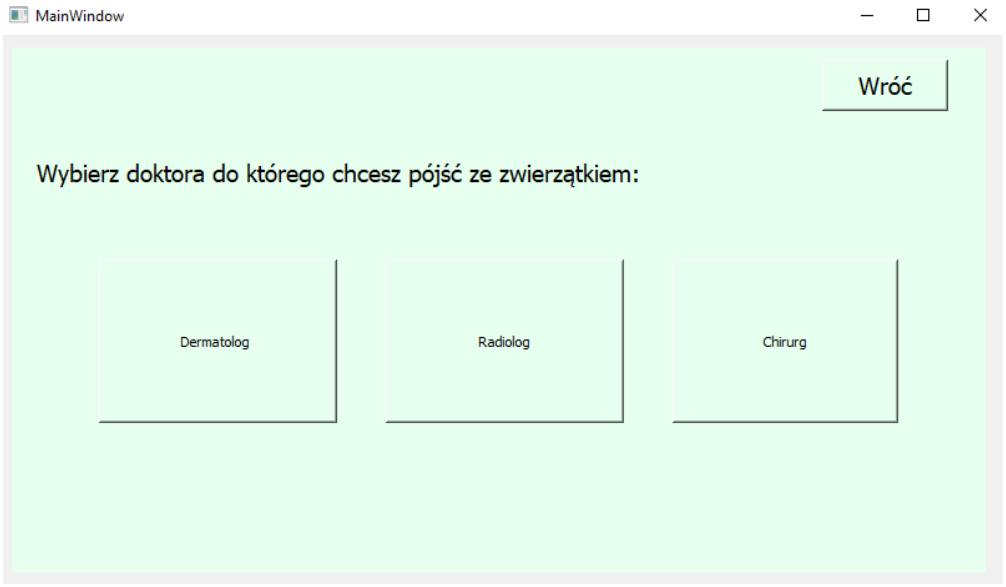
Podobny widok pokaże nam się, gdy klikniemy przycisk „Zabawa”. W polu po prawej stronie możemy wybrać długość trwania zabawy w godzinach używając strzałek lub wpisując wartość od 1 do 5, a następnie podwójnym kliknięciem należy wybrać zabawę z listy po lewej stronie. Należy wybierać odpowiednie zabawy dla naszego typu zwierzaka. W głównym oknie aplikacji pojawi się animacja.



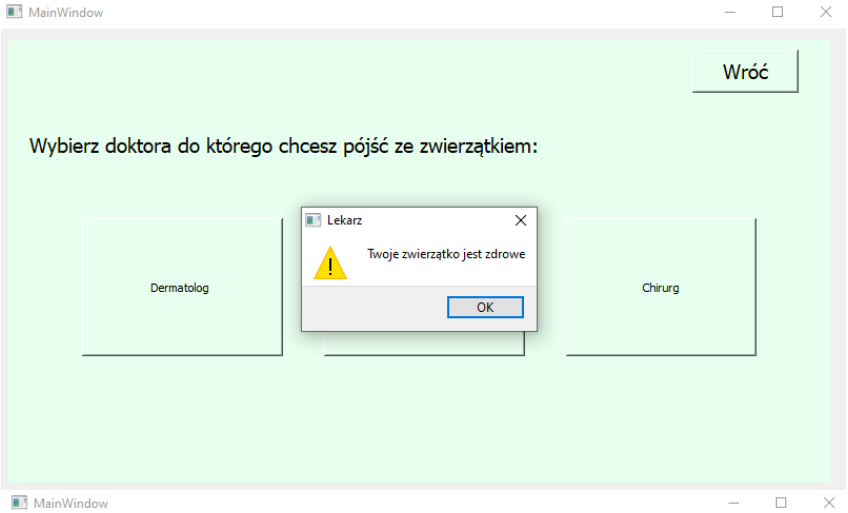
Wybierając przycisk „Sen” mamy możliwość położenia spać naszego zwierzaka. Przesuwając suwakiem wybieramy czas snu, który wyświetla nam się po prawej stronie.

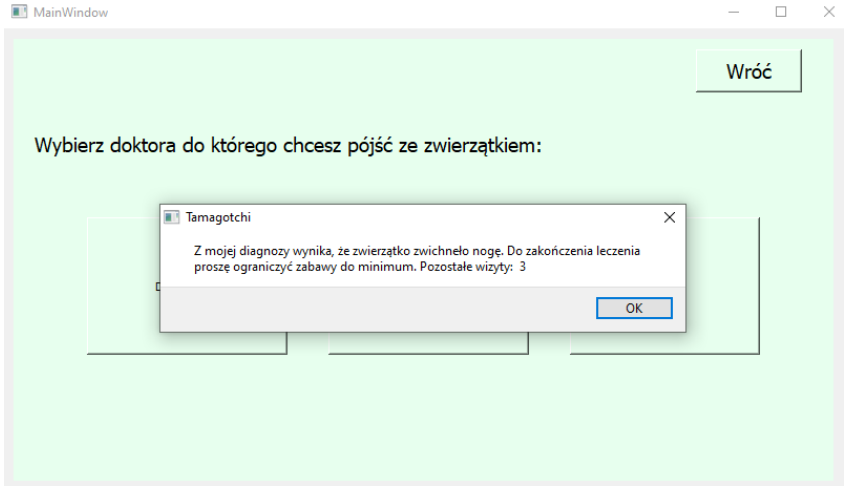


Zauważając pewne anomalie podczas opieki nad zwierzątkiem być może będziemy musieli udać się z nim do lekarza. Można to uczynić wybierając przycisk „Wizyta u doktora”, a następnie z widoku wybrać odpowiadającego nam lekarza.



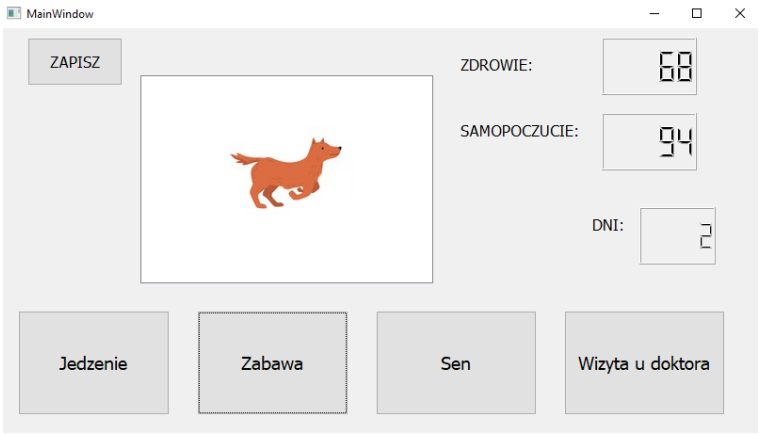
Po wybraniu doktora otrzymamy od niego informację o stanie zdrowia naszego zwierzątka oraz dalsze instrukcje. Danego dnia możemy odbyć tylko jedną wizytę.

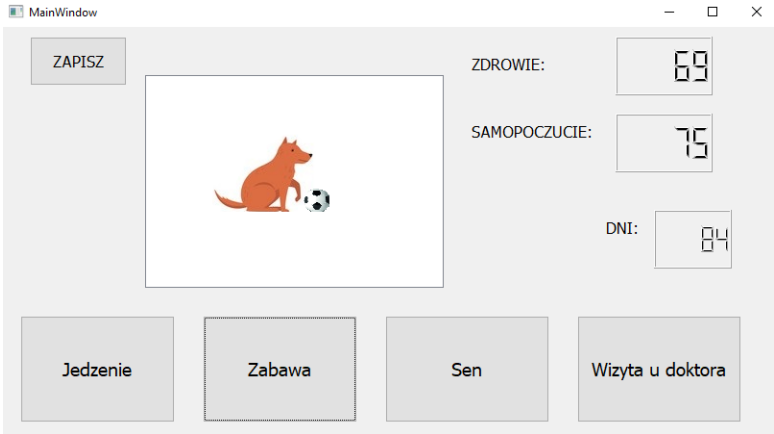




Gdy wskaźnik zdrowia spadnie do zera pojawia się widok końca gry. Użytkownik może zmienić konto lub utworzyć nowe zwierze na swoim koncie.

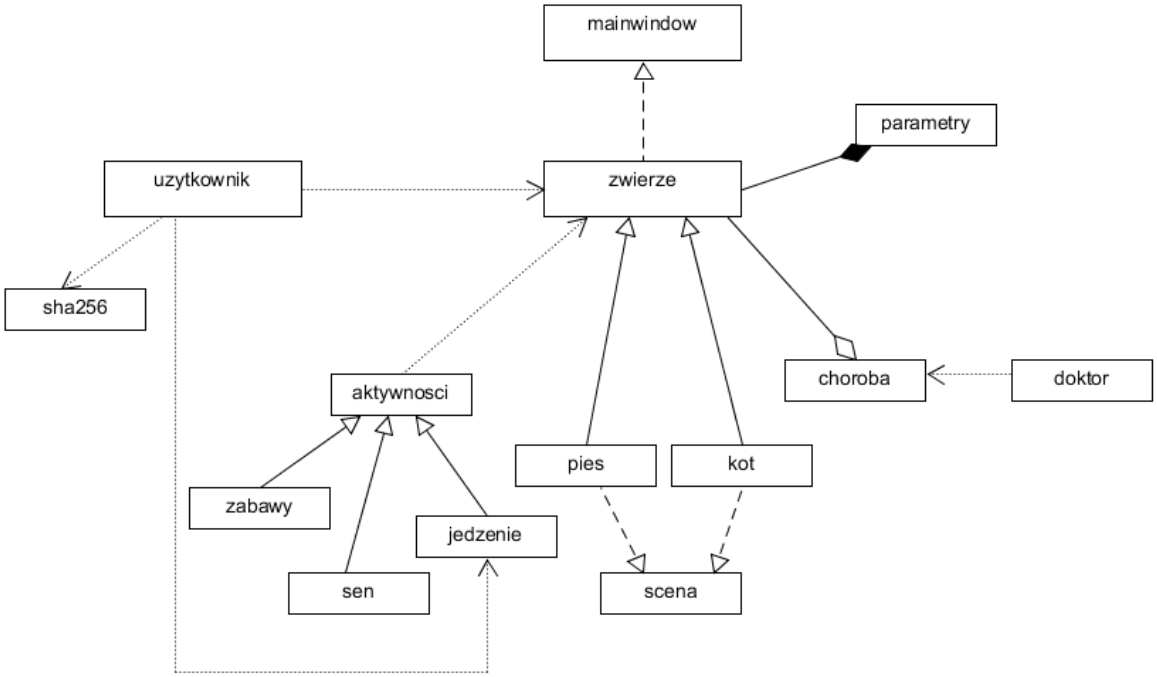
Przykładowe widoki gry:





4. Specyfikacja wewnętrzna.

Diagram klas:



Klasy:

- **mainwindow**
  Klasa odpowiedzialna za tworzenie okna oraz wszystkich elementów, takich jak przyciski, widżety list, pola do edycji, które się w nim znajdują. Jako składniki prywatne zadeklarowane są używane w trakcie trwania programu wskaźniki, listy oraz obiekty. Zawiera metody „nazwaprzycisku_onclick" które odpowiedzialne są najczęściej za walidację danych a następnie wywołanie odpowiednich metod na obiekcie użytkownika lub zwierzęciu. W konstruktorze tworzone są listy użytkowników, listy zabaw, a także ustawienia stałych elementów widoku.
  <u>Metody (wybrane):</u>
  on_button_zaloguj_clicked – po podaniu przez użytkownika prawidłowych danych wywołuje kolejne metody odtwarzającego jego poprzedni stan gry. Uruchamia metodę zliczającą upływ czasu na nowym wątku.
  on_button_nowe_konto_clicked oraz on_button_ng_pies/kot_clicked() – po podaniu przez nowego użytkownika nicku i hasła spełniających wymagania (sprawdzane regexem) wywołuje metody tworzące nową postać oraz tworzy listę jedzenia. Uruchamia metodę zliczającą upływ czasu na nowym wątku.
  on_list_widget_zabawy/jedzenie_itemDoubleClicked – wywołują metody bawsie/zjedz uwzględniając wybór użytkownika
  on_button_zatw_posilek_clicked – sprawdza czy dane nowego posiłku wprowadzone przez użytkownika są poprawne i dodaje posiłek od menu.

- **uzytkownik**
  Klasa zawierająca pola string (nazwa_uzytkownika, haslo, nawa_pliku_menu) identyfikujące danego użytkownika. Podczas uruchamiania programu plik z wszystkimi użytkownikami jest odczytywany i zapisywany do listy, co pozwala przeszukiwać liste w celu potwierdzenia hasła podczas logowania.
  Metody:
  odczytaj_plik_uzytkownika – odczytuje plik z informacjami o zwierzątku użytkownika zwracając vector z danymi (dane ułożone w kolejności w jakiej zostały zadeklarowane w klasach) wykorzystywanymi w konstruktorze zwierzęcia.
  zapisz_plik_uzytkownika – zapisuje do pliku o nazwie takiej samej jak nazwa użytkownika informacje o zwierzęciu, po kliknięciu „Zapisz" lub zamknięciu okna, wykorzystując przeciążony operator wypisania obiektu do strumienia.
  odczytaj/zapisz plik_menu – odczytuje/zapisuje informacje o menu użytkownika podczas logowania/zapisywania gry.
  dodaj_menu_uz – po podaniu nazwy użytkownika pozwala odczytać jego plik z posiłkami oraz dodać menu do aktualnie zalogowanego użytkownika.
  dodaj_posilek – pozwala dodać użytkownikowi posiłek do jego menu
  usun_zwierze – czyści plik z informacjami o zwierzęciu po przegranej grze

- **zwierze**
  Klasa ta dziedziczy z klas QObject oraz QGraphicsPixmapItem, aby umożliwić animowanie akcji. Zawiera pola z losowanymi cechami zwierzęcia (senność, głodomór, na_zewnątrz) które wpływają na to jak zwierze reaguje na daną czynność; wskaźnik na obiekt klasy parametry oraz choroba; wskaźniki na QTimer'y wywołujące odpowiednie metody z animacjami.
  Metody:
  Zawiera konstruktory przeciążone, które są wywoływane w zależności od tego czy tworzymy nowe zwierze czy odtwarzamy stan gry użytkownika.
  Zawiera metody czysto wirtualne, które są przesłaniane w klasach pies i kot.

- **pies**
  Klasa ta dziedziczy z klasy zwierze, a jej głównym zadaniem jest wyświetlanie animacji w głównym widoku aplikacji. Zawiera pola QPixmap pozwalające wgrać obrazy imitujące animacje oraz kilka pól int używanych do sterowania nimi.
  Metody:
  aktywuj_zabawa/sen/jedzenie – metody odpowiedzialne za uruchomienie zegara QTimer, aby ten przez wyznaczony czas wywoływał funkcje animujące.
  aktywuj_animacje_zabawy/snu/jedzenia – metody wywoływane przez przypisany im QTimer zmieniające odpowiednie obrazki QPixmap, aby imitować animacje.

- **kot**
  Klasa ta dziedziczy z klasy zwierze, a jej głównym zadaniem jest wyświetlanie animacji w głównym widoku aplikacji. Zawiera pola QPixmap pozwalające wgrać obrazy imitujące animacje oraz kilka pól int używanych do sterowania nimi.
  Metody:
  aktywuj_zabawa/sen/jedzenie – metody odpowiedzialne za uruchomienie zegara QTimer, aby ten przez wyznaczony czas wywoływał funkcje animujące.
  aktywuj_animacje_zabawy/snu/jedzenia – metody wywoływane przez przypisany im QTimer zmieniające odpowiednie obrazki QPixmap, aby imitować animacje.

- **scena**
  Klasa dziedzicząca z klasy QGraphicsScene potrzebna jedynie do ustawienia obiektu graphicsView, aby możliwe było dodanie do niej obrazków.

- **parametry**
  Klasa zawierająca pola informujące o statystykach zwierzęcia takich jak zdrowie, samopoczucie oraz ilość dni i godzin, a także poziomy głodu i snu, informacje o odbytych danego dnia aktywnościach, które wypływają poztywnie lub negatywnie na wskaźniki zdrowia oraz samopoczucia.
  Metody:
  zlicz15s – uruchamiana na nowym wątku, zlicza poszczególne godziny podczas trwania gry użytkownika na podstawie aktualnego czasu komputera oraz zdba o to, aby wartości na wskaźnikach były zawsze aktualne, a także w zależności od poziomów głodu/snu/odbytych aktywności odejmuje odpowiednie wartości od wskaźników zdrowia oraz samopoczucia. Oblicza pradopodobieństwo wystąpienia choroby u zwierzaka.
  aktualizujZS – metoda zwiększa lub zmniejsza wskaźniki zdrowia oraz samopoczucia o przekazane wartości, dbając aby nie przekroczyły wartości granicznych. Monitoruje czy poziom zdrowia nie spada do 0, jeśli spadnie zmienia widok w grze na końcowy.
  obnizglodisen – metoda zwiększa lub zmniejsza wskaźniki snu oraz glodu o przekazane wartości, dbając aby nie przekroczyły wartości granicznych.
  dodaj_czas_akcji – dodaje czas trwania zabaw/snu do czasu życia zwierzaka.

- **aktywnosci**
  Klasa zawiera podstawowe informacje o aktywności takie jak nazwa oraz o ile obiekt jest w stanie zregenerować zdrowie oraz samopoczucie.

- **jedzenie**
  Klasa dziedzicząca z klasy aktywnosci, zawiera podstawowe informacje o danym jedzeniu. Obiekty tej klasy tworzone są po uzyskaniu informacji o nich z pliku i umieszczane w liście oraz QWidgetList, aby umożliwić użytkownikowi wybór danego rekordu. W konstruktorze obliczany jest poziom regeneracji zdrowia oraz samopoczucia, ponieważ użytkownik może dodać swój obiekt.

Metody:

zjedz() - metoda wywoływana na rzecz obiektu wybranego przez użytkownika. W zależności od przeznaczenia jedzenia, kaloryczności oraz stanu zdrowia zwierzaka oblicza odpowiednie wartości które należy uwzględnić we wskaźnikach zdrowie oraz samopoczucie.

- **zabawy**
  Klasa dziedzicząca z klasy aktywnosci, zawiera podstawowe informacje o zabawie. Obiekty tej klasy tworzone sa podczas uruchamiania programu i umieszczane w liście oraz QWidgetList, aby umożliwić użytkownikowi wybór danego rekordu.
  Metody:
  baw_sie – metoda wywoływana na rzecz obiektu wybranego przez użytkownika. W zależności od stanu zdrowia zwierzaka, typu zabawy, długości trwania przelicza odpowiednie wartości które należy uwzględnić we wskaźnikach zdrowie oraz samopoczucie.

- **sen**
  Klasa dziedzicząca z klasy aktywnosci.
  Metody:
  spij – uwzględniając wybrany przez użytkownika czas regeneruje wskaźniki zdrowia oraz samopoczucia

- **choroba**
  Klasa zawierająca podstawowe informacje o danej chorobie (nazwa, ograniczenia, odp_lekarz, minus_zdrowie, wymagane_wizyty).
  Posiada konstruktory przeciążone, które są wywoływane w zależności od tego czy ma zostać utworzona nowa choroba, czy przy logowaniu odczytujemy informacje o istnejącej już chorobie. Typ choroby, który ma zostać przydzielony zwierzakowi jest losowany.

- **doktor**
  Klasa zawierająca podstawowe informacje o lekarzu (imię, nazwisko, specjalizacja) oraz wektor jego diagnoz i wskazówek dla opiekuna zwierzaka, gdy ten zachoruje.
  Metody:
  stworz_dok – metoda pozwala nam stworzyć doktora z charakterystycznymi dla niego diagnozami
  wizyta – metoda sprawdzająca stan choroby zwierzaka, wyświetla informacje od doktora w wyskakującym oknie oraz pozwala poprawić wskaźniki zdrowia oraz samopoczucia zwierzaka. Danego dnia można odbyć jedną wizytę.

Wykorzystane techniki obiektowe:

- polimorfizm (w klasach zwierze, kot i pies dla ułatwienia wprowadzania nowych zwierząt do aplikacji oraz ułatwienia wywoływania metod w programie głownym dzięki wiązaniom dynamicznym)

- regex (do sprawdzania poprawności wprowadzanego hasła)

- kontenery STL (dla uporządkowywania obiektów klas jedzenie, zabawy oraz użytkownicy; używanie wektorów do przesyłania dużej ilości informacji pomiędzy klasami)

- algorytmy STL (przeszukiwanie list, sortowanie w celu znalezienia rekordów wybranych przez użytkownika; iteratory)

- wątki (wywoływanie funkcji zliczającej czas podczas trwania gry na osobnym wątku)

- inteligentne wskaźniki (tworzenie nowych obiektów kot i pies, użytkownik; zapewniają odpowiednie zwalnianie pamięci podczas przelogowywania oraz końca gry)