AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INŻYNIERII METALI I INFORMATYKI PRZEMYSŁOWEJ

KATEDRA *Informatyki Stosowanej i Modelowania*

Projekt dyplomowy

Opracowanie oraz implementacja aplikacji desktopowej pomagającej w nauce elektroniki
Development and implementation of a desktop application helping in learning electronics

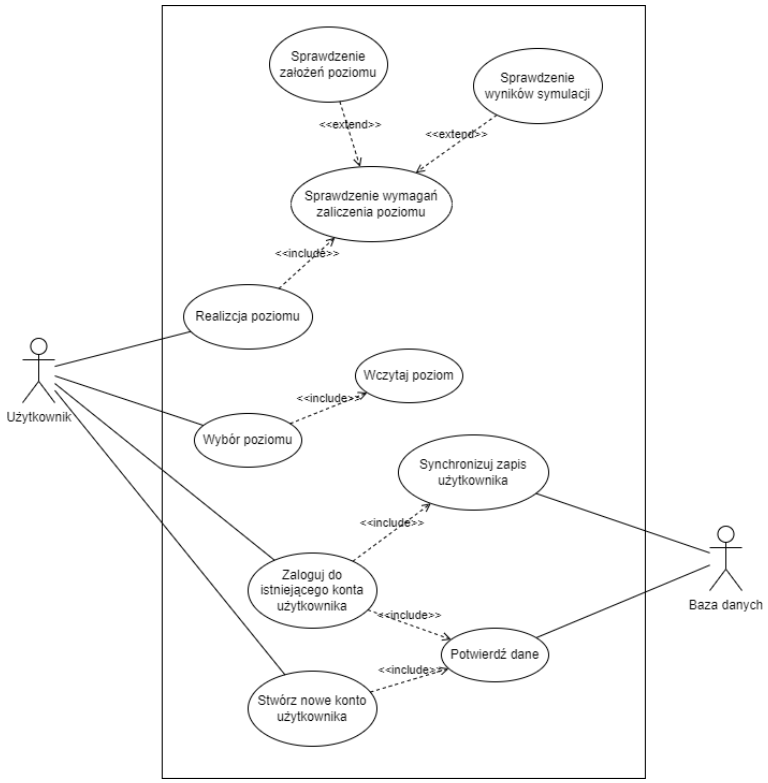
Autor: *Bartłomiej Gojowczyk*
Kierunek studiów: *Informatyka Techniczna*
Opiekun projektu: *dr inż. Piotr Kustra*

Kraków, *2024*

1 Idea projektu/Wstęp

2 Idea działania aplikacji

Działanie aplikacji polega na realizowaniu kolejnych poziomów. Każdy poziom zawiera opis działania układu oraz opis połączenia komponentów elektronicznych. Użytkownik ma możliwość zarejestrowania oraz zalogowania się do aplikacji w celu zapisania postępu w ”chmurze”. Na rys. 1 przedstawiony został diagram przypadków użycia, który pokazuje funkcjonalne aspekty aplikacji.



Rys. 1: Diagram przypadków użycia

Źródło: opracowanie własne

Pytanie 1: moze dodać opis każdego use casa?

3 Projekt aplikacji

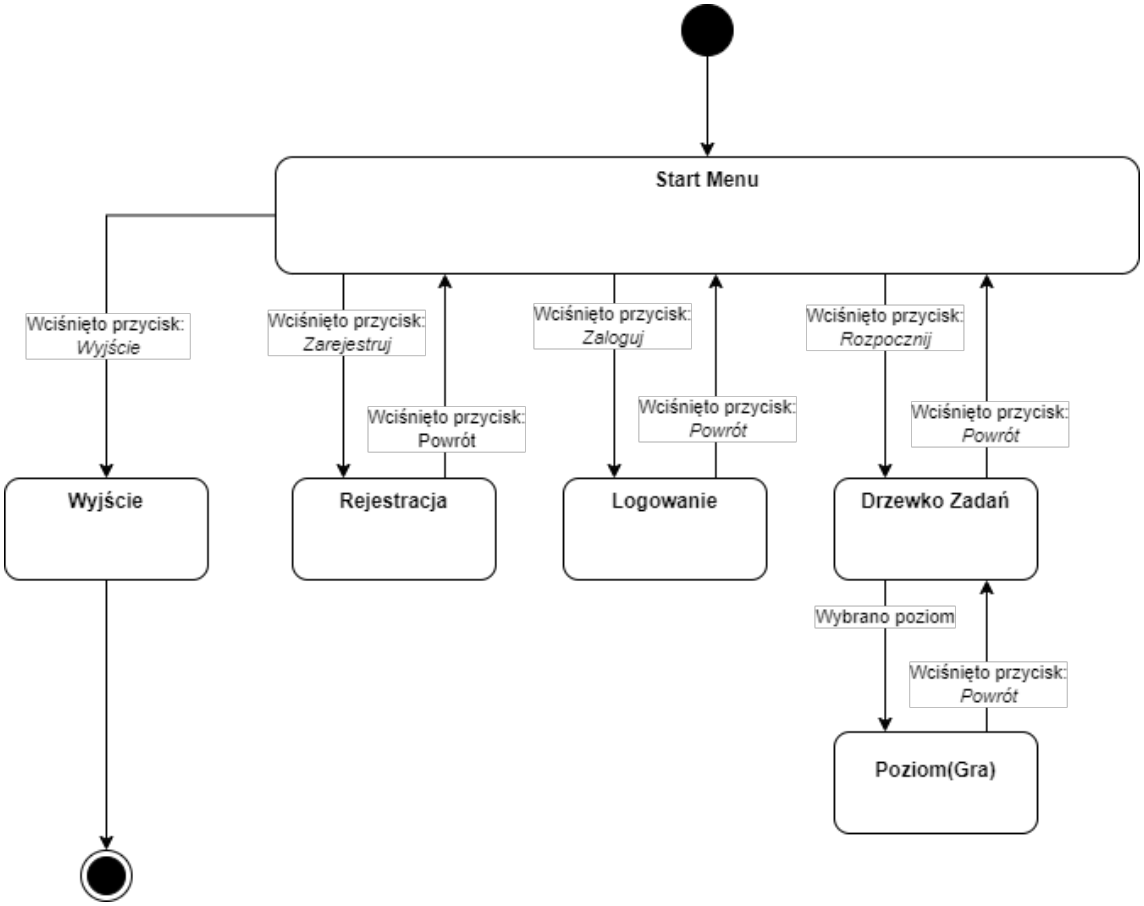
Procesor wykonuje instrukcje w sposób sekwencyjny, więc do zachowania ciągłości renderowania obrazu wymagane jest, by aplikacja zawierała nieskończoną pętlę, w której jest zawarte sprawdzanie zdarzeń, aktualizowanie aplikacji oraz rysowanie okna [1]. Schemat blokowy głównej pętli został przedstawiony na rys. 2. Dodany warunek sprawdzania fokusu okna pozwala na nie wykorzystywanie zasobów sprzętowych, w przypadku gdy okno jest nie aktywne, bądź jest zminimalizowane.

Rys. 2: Schemat blokowy pętli głównej

Źródło: opracowanie własne na podstawie [1]

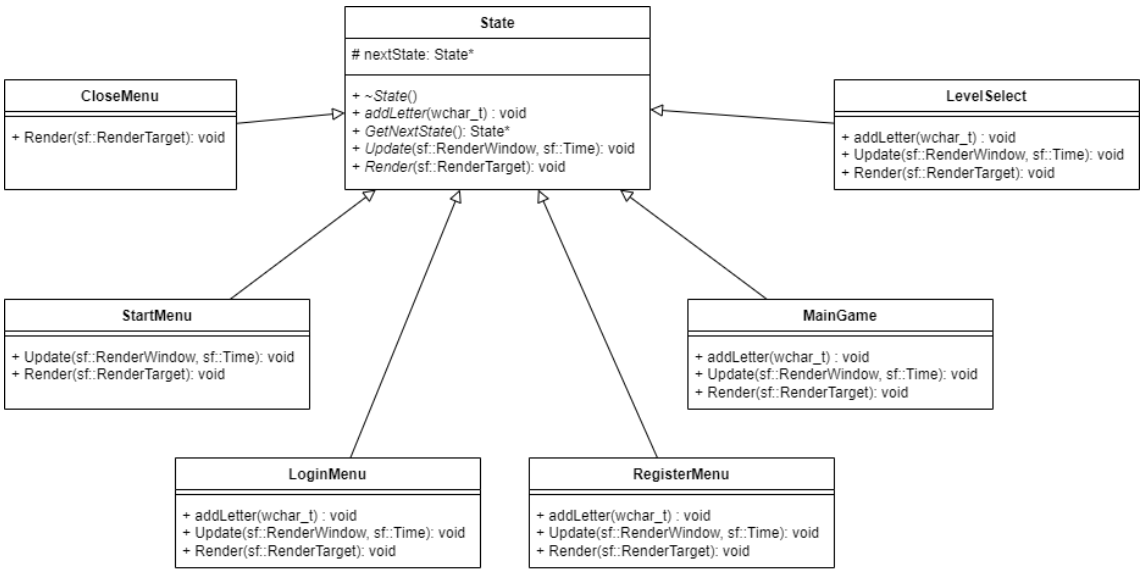
Przechodzenie pomiędzy widokami aplikacji zostało zrealizowane w klasie *Game*. Diagram maszyny stanów tej klasy został przedstawiony na rys. 3. Pytanie 2: w jaki sposób powinienem odwoływać się do nazw klas/obiektów, kursywą?

Gdy użytkownik ułoży komponenty na planszy, a następnie wciśnie przycisk sprawdzający zadanie, to aplikacja utworzy plik z schematem. Następnie aplikacja wykona symulację wykorzystując zewnętrzny, darmowy program **LTspice** oraz wygenerowany plik. Rys. 5 przedstawia diagram sekwencji zawierający interakcję między obiektami, gdy aplikacja będzie sprawdzała poziom.



Rys. 3: Diagram maszyny stanów obiektu klasy *Game*

Źródło: opracowanie własne



Rys. 4: Uproszczony diagram klas pokazujący dziedziczenie klas dla maszyny stanów obiektu klasy *Game*

Źródło: opracowanie własne

Rys. 5: Diagram sekwencji, pokazujący interakcję podczas symulacji

Źródło: opracowanie własne

4 Implementacja aplikacji

4.1 Wprowadzenie

Główna część aplikacja została zaimplementowana w języku C++, wykorzystując standard ISO C++ 14. Do komunikacji z zewnętrznym programem wykorzystany został język Python w wersji 3.12 osadzony w głównej części aplikacji. Takie rozwiązanie pozwala na bezpośredni dostęp do zasobów wykonywanego kodu w języku Python przez aplikajcę w C++. Aplikacja została zaimplementowana z myślą o systemie operacyjnym Windows, natomiast wykorzystane biblioteki pozwalają na przeniesie na inne systemy operacyjne dystrybucji Linux czy macOS. Pytanie 3: not sure

Wykorzystaną bazą danych jest MySQL.

4.2 Środowisko oraz narzędzia programistyczne

Wykorzystanym środowiskiem programistycznym było Visual Studio 2022. Zostały wykorzystane moduły: *Graphics, System, Window* biblioteki SFML *(Simple and Fast Multimedia Library)* opartej o OpenGL. Wykorzystanie tych modułów pozwala na proste i wydajne rysowanie okna na ekranie oraz przekazywanie własnych parametrów do strumienia przetwarzania potoku grafiki. Zastosowana została również standardowa biblioteka STL.

Aplikacja w celu przekazania i odebrania danych z symulacji wykorzystuje bibliotekę Python. Natomiast w zagnieżdżonej części Python wykorzystuje moduł PyLTSpice do realizacji symulacji oraz moduł ltspice do odczytania obliczonych wartości symylacji.

Do wykonywania zapytań z bazą danych wykorzystana została biblioteka mysqlx. Baza danych była uruchamiana w środowisku kontenerowym Docker, do inicjalizacji tabel oraz relacji wykorzystany został język SQL.

ciekawsze mi się wydaje: - pakowanie/rozpakowywanie danych z db - coś z poziomem - fragmenty z symulacji - jak wygląda płytka - dodawanie ścieżki - przykładowy poziom

```
  class PopupBox : public sf::Drawable
{
public:
  PopupBox()
  {

  }

private:
  virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const
```

```
11   {
12     target.draw(text, states);
13   }
14
15   sf::Font* font;
16   sf::Text text;
17 };
```

Fragment kodu 1: awawd

Źródło: opracowanie własne

5 Uruchomienie aplikacji

6 Testowanie aplikacji

Literatura

[1] Game Programming Patterns, Robert Nystrom, https://gameprogrammingpatterns.com/game-loop.html