Zgadywanka - Gra z Obrazkami

Sprawozdanie Projektowe

Julia Hryciuk, Mateusz Drozd, Jakub Kalita

27 listopada 2024

Spis treści

1 Wstęp

W dzisiejszych czasach tworzenie gier komputerowych staje się coraz bardziej popularne i dostępne. Celem tego projektu było stworzenie prostej, interaktywnej gry w języku C++, żeby rozwijać umiejętności praktyczne w zakresie programowania.

2 Streszczenie

Projekt "Zgadywanka"to gra polegająca na odgadywaniu obrazków. Na planszy ukazane jest 12 elementów, z których co pewien czas odsłania się jeden. Gracz musi zgadnąć, co przedstawia obrazek, korzystając z okna odpowiedzi. Po udanej odpowiedzi wyświetla się komunikat o poprawnej odpowiedzi, a gracz może rozpocząć nową grę z losowym obrazkiem. Po niepoprawnej odowiedzi wyswietla sie komunikat o błędzie, a gracz musi zgadywać dalej. Gra została zaimplementowana w środowisku Code::Blocks, wykorzystując podstawy programowania obiektowego.

3 Cel Projektu

Główne cele projektu obejmowały:

- Implementację interaktywnej planszy z elementami.
- Losowe odsłanianie obrazków co kilka sekund.
- Możliwość udzielenia odpowiedzi przez gracza.
- Obsługę poprawnych i błędnych odpowiedzi.
- Losowanie nowych obrazków po poprawnej odpowiedzi.
- Ustawienie czasu, aby gracz wiedział jak szybko udaje mu sie podac prawidłową odpowiedź.

4 Struktura Katalogów: Zgadywanka

Struktura katalogów projektu Zgadywanka jest zorganizowana w następujący sposób:

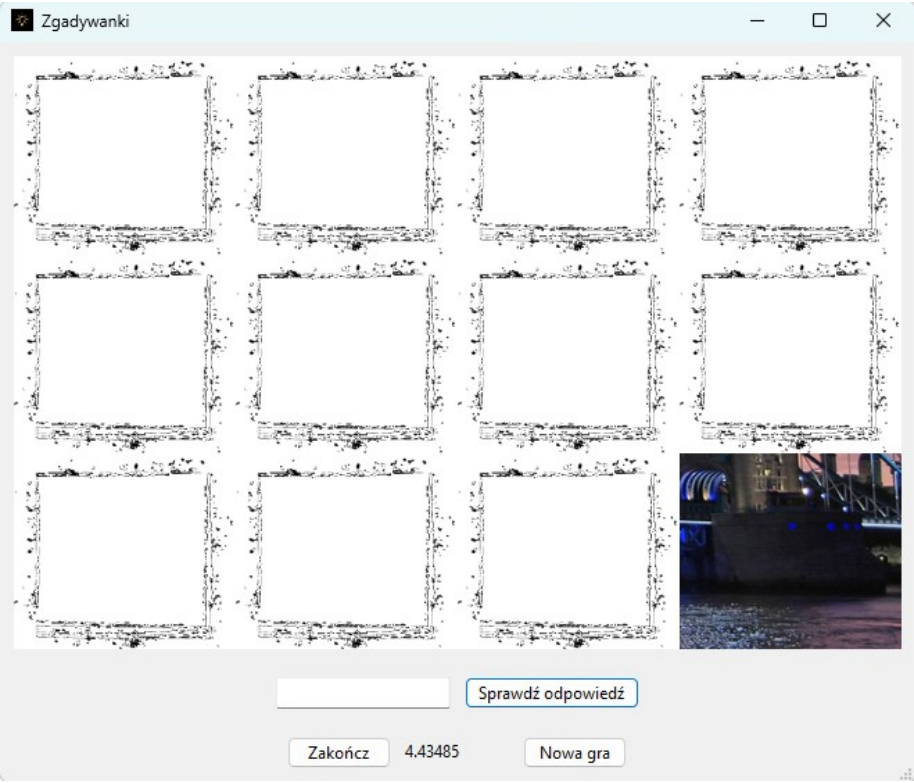
- `Zgadywanka/`
  - `src/` - Kod źródłowy aplikacji.
  - `images/` - Zasoby graficzne (obrazki).
  - `docs/` - Dokumentacja projektu.
  - `install/` - Pliki instalacyjne.

5 Opis Aplikacji

Aplikacja prezentuje interaktywną planszę z elementami, z których jeden co jakiś czas odsłania się na krótko. Gracz musi odgadnąć, co przedstawia obrazek, wpisując odpowiedź w oknie przewidzianym do tego celu. Po poprawnej odpowiedzi wyświetla się komunikat, a gracz ma możliwość rozpoczęcia nowej gry z losowym obrazkiem. W przypadku błędnej odpowiedzi czas się nie zatrzymuje, a gracz może próbować ponownie.

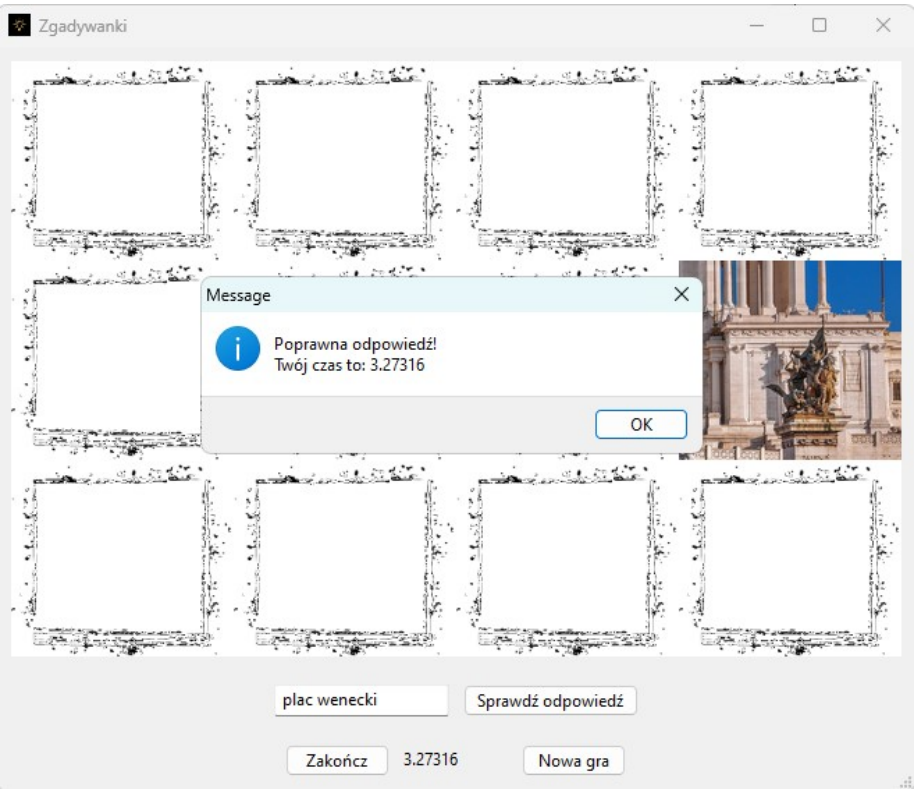
6 Jak wygląda gra

6.1 Podczas Gry



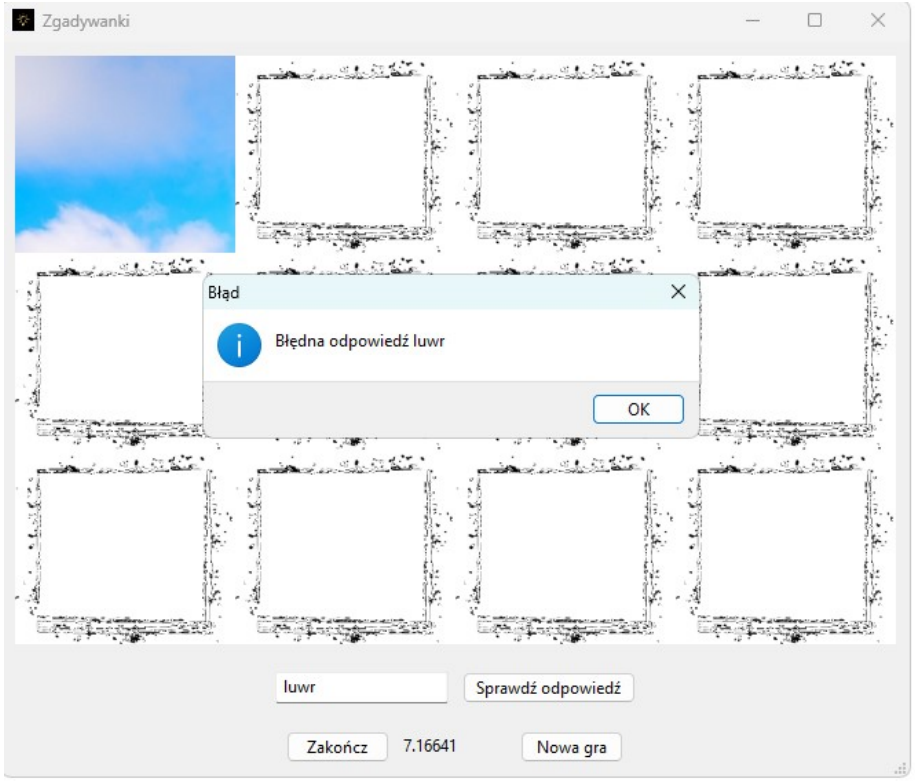
Rysunek 1: Zrzut ekranu podczas trwania gry.

6.2 Po Poprawnej Odpowiedzi



Rysunek 2: Zrzut ekranu po udzieleniu poprawnej odpowiedzi.

6.3 Po Błędnej Odpowiedzi



Rysunek 3: Zrzut ekranu po udzieleniu błędnej odpowiedzi.

7 Implementacja

7.1 Implementację interaktywnej planszy z elementami

Poniżej przedstawiono fragment kodu odpowiedzialnego za odsłanianie elementów na planszy:

```
    wxStaticBitmap* pola[16];
    wxBitmap rysunki[4];
    std::map <int,int> id2nr;
    zgadywanka gra;
    int licznik=0;
    int numer_obrazka = (rand() % 17);
    int b=numer_obrazka;

    std::string nazwa_obrazka =
       "zdjecia/"+std::to_string(numer_obrazka+1)+".png";
```

```
10
11      wxBitmap zakryj(int k = -1);
12
13
14      wxBitmap obrazek, obrazek_zakryty;
```

```
1       wxBitmap ProjectDialog::zakryj(int k) {
2         wxBitmap tmp = obrazek;
3         wxMemoryDC dc;
4         dc.SelectObject(tmp);
5         for (int i = 0; i < 16; ++i) {
6           if (k==i)
7           continue;
8           int x = (i % 4) * 160;
9           int y = (i / 4) * 143;
10          dc.DrawBitmap(wxImage("kwadracik.png"), wxPoint(x, y));
11        }
12        dc.SelectObject(wxNullBitmap);
13        return tmp;
14      }
```

7.2 Losowe odsłninie obrazów co 50 sekund

Poniżej przedstawiono fragment kodu odpowiedzialnego za losowe odsłninie obrazów co kilka sekund :

```
1       void ProjectDialog::OnTimer1Trigger(wxTimerEvent& event)
2       {
3         licznik++;
4         double c = licznik * Timer1.GetInterval() / 1274.0;
5         wxString czas;
6         czas << c;
7         StaticText1->SetLabel(czas);
8         if(licznik ==6368){
9           Timer1.Stop();
10          Timer2.Stop();
11          wxString answersFilePath = wxT("Rozwiazania.txt"); // Zmie'n
                na wła'sciwą 'scie.zkę do pliku
12          wxTextFile file;
13          if (file.Open(answersFilePath)) {
14
15            // Sprawd'z, czy numer linii odpowiada numerowi obrazka
16
17
18            wxString correctAnswer = file.GetLine(numer_obrazka);
19            correctAnswer = correctAnswer.Lower();
```

```
20
21          // Pobierz wpisaną przez u.zytkownika odpowied'z
22          wxString enteredText = TextCtrl1->GetValue();
23          enteredText = _(enteredText.Lower());
24          // Por'ownaj odpowied'z z poprawną odpowiedzią z pliku
25          wxMessageBox(_("Nowy obrazek zostanie wczytany \nTw'oj
                czas to "+ czas));
26          numer_obrazka = (rand() % 17);
27          while(b==numer_obrazka){
28
29            numer_obrazka = (rand() % 17);
30          }
31          std::string nazwa_obrazka =
                "zdjecia/"+std::to_string(numer_obrazka+1)+".png";
32          StaticBitmap2->SetBitmap(wxImage(nazwa_obrazka));
33          obrazek = StaticBitmap2->GetBitmap();
34          obrazek_zakryty = zakryj();
35          StaticBitmap2->SetBitmap(obrazek_zakryty);
36          TextCtrl1->Clear();
37          licznik=0;
38          Timer2.Start();
39          Timer1.Start();
40
41        }
42      }
43    }
```

7.3 Możliwość udzielenia odpowiedzi przez grcza

Poniżej przedstawiono fragment kodu odpowiedzialnego za możliwość udzielenia odpowiedzi przez grcza:

```
1   void ProjectDialog::OnTimer2Trigger(wxTimerEvent& event)
2   {
3     int k = rand() % 12;
4     auto obrazek = zakryj(k);
5     StaticBitmap2->SetBitmap(obrazek);
6   }
```

7.4 Obsługa poprawnyh i błędnych opowiedzi

Poniżej przedstawiono fragment kodu odpowiedzialnego za obsługa poprawnyh i błędnych opowiedzi:

```
1     void ProjectDialog::OnButton2Click(wxCommandEvent& event)
2
```

```
3     {
4
5       // Utw'orz nazwę pliku tekstowego zawierającego odpowiedzi
6       wxString answersFilePath = wxT("Rozwiazania.txt"); // Zmie'n
            na wła'sciwą 'scie.żkę do pliku
7       Timer1.Stop();
8       Timer2.Stop();
9       // Otw'orz plik tekstowy
10      double c = licznik * Timer1.GetInterval() / 1274.0;
11      wxString czas;
12      czas << c;
13      wxTextFile file;
14
15      if (file.Open(answersFilePath)) {
16
17        // Sprawd'z, czy numer linii odpowiada numerowi obrazka
18
19
20        wxString correctAnswer = file.GetLine(numer_obrazka);
21        correctAnswer = correctAnswer.Lower();
22
23        // Pobierz wpisaną przez u.zytkownika odpowied'z
24
25        wxString enteredText = TextCtrl1->GetValue();
26        enteredText = enteredText.Lower();
27        // Por'ownaj odpowied'z z poprawną odpowiedzią z pliku
28
29        if (enteredText == correctAnswer) {
30          wxMessageBox(_("Poprawna odpowied'z! \nTw'oj czas to:
                "+czas));
31
32          wxMessageBox("Nowy obrazek zostanie wczytany");
33          numer_obrazka = (rand() % 17);
34          while(b==numer_obrazka){
35
36            numer_obrazka = (rand() % 17);
37          }
38          std::string nazwa_obrazka =
                "zdjecia/"+std::to_string(numer_obrazka+1)+".png";
39
40          wxString correctAnswer = file.GetLine(numer_obrazka);
41          correctAnswer = correctAnswer.Lower();
42
43          StaticBitmap2->SetBitmap(wxImage(nazwa_obrazka));
44          obrazek = StaticBitmap2->GetBitmap();
45          obrazek_zakryty = zakryj();
```

```
46            StaticBitmap2->SetBitmap(obrazek_zakryty);
47            licznik=0;
48            Timer1.Start();
49            Timer2.Start();
50          }
51
52          else {
53
54            wxMessageBox(_("Błędna odpowied'z: "+ enteredText),
                 _("Błąd"));
55            Timer1.Start();
56            Timer2.Start();
57          }
58
59
60
61          TextCtrl1->Clear();
62            }
63          }
```

7.5 Losowanie nowyh obrazków po udzieleniu poprawnej odpowiedzi

Poniżej przedstawiono fragment kodu odpowiedzialnego za losowanie nowyh obrazków po udzieleniu poprawnej odpowiedzi:

```
1       if (enteredText == correctAnswer) {
2         wxMessageBox(_("Poprawna odpowied'z! \nTw'oj czas to: "+czas));
3
4         wxMessageBox("Nowy obrazek zostanie wczytany");
5         numer_obrazka = (rand() % 17);
6         while(b==numer_obrazka){
7
8           numer_obrazka = (rand() % 17);
9         }
10        std::string nazwa_obrazka =
             "zdjecia/"+std::to_string(numer_obrazka+1)+".png";
11
12        wxString correctAnswer = file.GetLine(numer_obrazka);
13        correctAnswer = correctAnswer.Lower();
14
15        StaticBitmap2->SetBitmap(wxImage(nazwa_obrazka));
16        obrazek = StaticBitmap2->GetBitmap();
17        obrazek_zakryty = zakryj();
18        StaticBitmap2->SetBitmap(obrazek_zakryty);
19        licznik=0;
20        Timer1.Start();
```

```
21        Timer2.Start();
22    }
```

7.6 Ustawienie czasu, aby gracz wiedział jak szybko udaje mu sie podać poprawn odpowiedź oraz losowanie nowego obrazka po 50 sekundach gry

Poniżej przedstawiono fragment kodu odpowiedzialnego za ustawienie czasu, aby gracz wiedział jak szybko udaje mu sie podać poprawną odpowiedź:

```
1     void ProjectDialog::OnButton3Click(wxCommandEvent& event)
2     {
3       Timer1.Stop();
4       Timer2.Stop();
5       wxString answersFilePath = wxT("Rozwiazania.txt");
6       wxTextFile file;
7       wxMessageBox("Nowy obrazek zostanie wczytany");
8       numer_obrazka = (rand() % 17);
9       while(b==numer_obrazka){
10
11        numer_obrazka = (rand() % 17);
12      }
13      b=numer_obrazka;
14      std::string nazwa_obrazka =
           "zdjecia/"+std::to_string(numer_obrazka+1)+".png";
15      StaticBitmap2->SetBitmap(wxImage(nazwa_obrazka));
16      obrazek = StaticBitmap2->GetBitmap();
17      obrazek_zakryty = zakryj();
18      StaticBitmap2->SetBitmap(obrazek_zakryty);
19      TextCtrl1->Clear();
20      licznik = 0;
21      Timer1.Start();
22      Timer2.Start();
23
24
25
26    }
```

8 Program Instalacyjny

Aby zainstalować aplikację Zgadywanka, wykonaj następujące kroki:

1. Przejdź do katalogu `install/`.
2. Uruchom plik instalacyjny `install.sh`.

3. Postępuj zgodnie z instrukcjami na ekranie.

9 Problemy napotkane podczas tworzenia aplikacji

Podczas procesu tworzenia aplikacji "Zgadywanka"napotkaliśmy pewne wyzwania, które wymagały dodatkowej uwagi i rozważań. Poniżej przedstawiamy szczegółowy opis tych problemów.

9.1 Problemy z Kwadratami

1. **Przesłanianie Kwadratów:** W trakcie implementacji zauważyliśmy, że kwadraty, reprezentujące elementy gry, czasami nakładają się na siebie w sposób, który nie spełnia naszych oczekiwań estetycznych. To zjawisko wymagało dodatkowej uwagi w zakresie warstw i pozycjonowania kwadratów, aby osiągnąć pożądany efekt wizualny.

2. **Animacja Odsłaniania:** Kolejnym problemem było zapewnienie płynnej animacji odsłaniania kwadratów. Choć podstawowa logika animacji została zaimplementowana, pojawiły się wyzwania związane z optymalizacją tego procesu, aby uzyskać satysfakcjonujący efekt wizualny.

9.2 Problemy z Czytaniem Małych i Dużych Liter

1. **Weryfikacja Wielkości Liter:** W funkcji odpowiedzialnej za odczyt liter, napotkaliśmy trudności z prawidłowym rozróżnieniem między małymi a dużymi literami. Było istotne stworzenie mechanizmu weryfikującego wielkość liter, aby poprawnie interpretować odpowiedzi użytkownika.

2. **Normalizacja Wielkości Liter:** Dodatkowym wyzwaniem było zagwarantowanie spójności w interpretacji liter bez względu na ich wielkość. Zastosowaliśmy funkcje dostępne w języku C++.

10 Podział pracy

- **Julia Hryciuk** .
- **Mateusz Drozd** .
- **Jakub Kalita** .

11 Podziękowania

Chcielibyśmy podziękować naszemu opiekunowi projektu za cenne wskazówki i wsparcie w trakcie realizacji zadania.

12 Bibliografia

- https://wiki.codeblocks.org/index.php/WxSmithtutorial:Helloworld
- https://fly.pl/aktualnosci/grecja-po-pozarach-bezpieczne-wakacje-w-grecji/
- https://pl.wikipedia.org/wiki/D
- https://motozwiedzanie.pl/palac-wenecki-klejnot-wiecznego-miasta/
- https://www.national-geographic.pl/traveler/artykul/luwr-ma-juz-230-lat-to-najchetniej-odwiedzane-muzeum-swiata
- https://www.unesco.pl/kultura/dziedzictwo-kulturowe/swiatowe-dziedzictwo/lista-swiatowego-dziedzictwa/