POLITECHNIKA
LUBELSKA
WYDZIAŁ MATEMATYKI
I INFORMATYKI TECHNICZNEJ

Kierunek: Inynieria i Analiza Danych

Sequence memory

Emil Szewczak

10 grudnia 2024

Spis treci

Wprowadzenie

Sequence memory to interaktywna aplikacja, w której gracze testuj i doskonal swoj zdolno do zapamitywania sekwencji podwietlanych kwadratów. Gra jest stworzona w jzyku C++ przy uyciu biblioteki wxWidgets, co umoliwia dynamiczny interfejs graficzny. Projekt ten by wzorowany na grze Sequence Memory ze strony https://humanbenchmark.com.

Cel Gry

Gra Sequence Memory stawia przed graczem zadanie zapamitywania i reprodukowania coraz bardziej wymagajcych sekwencji elementów, co rozwija zdolnoci poznawcze, takie jak spostrzeganie wzorców, koncentracja i zdolno do przewidywania. W trakcie rozgrywki uczestnik musi skupi uwag na szczegóach, doskonali umiejtno szybkiego rozpoznawania oraz utrzymywa informacje w pamici krótkotrwaej. To nie tylko dostarcza przyjemnoci z rozgrywki, ale take sprzyja treningowi umysowemu, co moe mie pozytywny wpyw na codzienne funkcje poznawcze.

Zasady Gry

- Test Sekwencji ocenia Twoj pami wzrokow i umiejtno powtarzania wzorców.
- Zadaniem jest powtórzenie sekwencji, któr zaprezentuje komputer. Sekwencje bd coraz dusze w kadej rundzie.
- Kliknij na pola w odpowiedniej kolejnoci, tak jak zostay przedstawione wczeniej. Upewnij si, e klikasz dokadnie w tej samej kolejnoci.
- Po powtórzeniu sekwencji, plansza zmieni kolor na zielony, jeli odpowied jest poprawna lub na czerwony, jeli popeniono bd.
- Zobaczysz równie komunikat o wyniku i rundzie na górze ekranu.
- Postaraj si pamita jak najdusze sekwencje, aby uzyska jak najwyszy wynik!
- Moesz wybra jeden z poród 3 poziomów trudnoci, rónicych si wielkoci planszy.
- Zdecydowa moesz te o iloci y, symbolizujcych liczb bdów, które moesz popeni. Gdy ich zabraknie gra si skoczy, a na ekranie pojawi si wynik.
- Kliknij przycisk „Start” i zacznij testowa swoj pami!

Elementy Interfejsu Graficznego

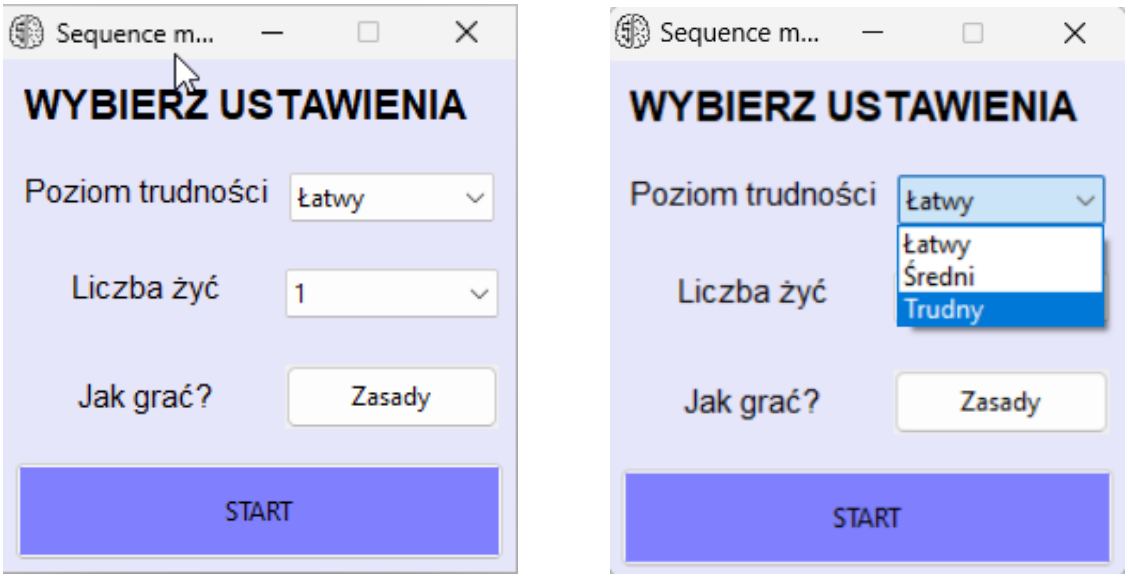
Interfejs graficzny gry skada si z dwóch gównych elementów: gównego menu oraz planszy gry, zaprojektowanych w jasny i przejysty sposób.

Gówne Menu

Gówne menu jest centralnym punktem interakcji z uytkownikiem. Oferuje ono moliwo dostosowania ustawie przed rozpoczciem gry. Skada si z czterech przycisków:

- Dwóch rozwijanych:
  - Poziom trudnoci - umoliwia wybór jednego z trzech poziomów trudnoci.
  - Liczba y - daje uytkownikowi wybór, co do iloci poziadanych y, podczas jednej rundy.
- Zasady - uruchamia reguy gry.
- Start - stanowi kluczowy przycisk w aplikacji, przenosi uytkownika na plansz oraz rozpoczyna rozgrywk.

Oprócz tego okno posiada moliwo zamknicia oraz zminimalizowania.(rysunek 1a)



(a) Gówne menu

(b) Wybór poziomu trudnoci

Rysunek 1: Menu gówne

Plansza Gry

Plansza gry jest obszarem, na którym odbywa si gówna rozgrywka. Jest ona tworzona dynamicznie, a jej wielko (ilo pól) zaley od poziomu trudnoci wybranego przez uytkownika. Skada si z kolorowych kwadratowych pól, które w poszczególnych etapach gry zostaj sekwencyjnie podwietlone. W górnej czci planszy znajduje si informacja dotyczca obecnej rundy oraz rysunki (serca), symbolizujce liczb y, jakie pozostay uytkownikawi. W dolnej czci natomiast zosta umieszczony przycisk „MENU", pozwalajcy wróci do menu gównego. Podobnie jak w przypadku okana Menu tu równie gracz ma moliwo zamknicia oraz zminimalizowania gry. (rysunek 2a).

Poziomy Trudnoci

Gra oferuje róne poziomy trudnoci, dajc graczom moliwo dostosowania wyzwania do swoich umiejtnoci i preferencji. Kady poziom trudnoci wpywa na rozmiar planszy:

- **Rozmiar Planszy:** W miar wzrostu poziomu trudnoci, plansza gry staje si wiksza, co wymaga od gracza wikszego skupienia uwagi i umiejtnoci lepszego zapamitywania.

  atwy poziom zawiera plansz rozmiarem 3x3 (rysunek 2a).

  redni poziom zawiera plansz rozmiarem 4x4 (rysunek 2b).

  Trudny poziom zawiera plansz rozmiarem 5x5 (rysunek 2c).

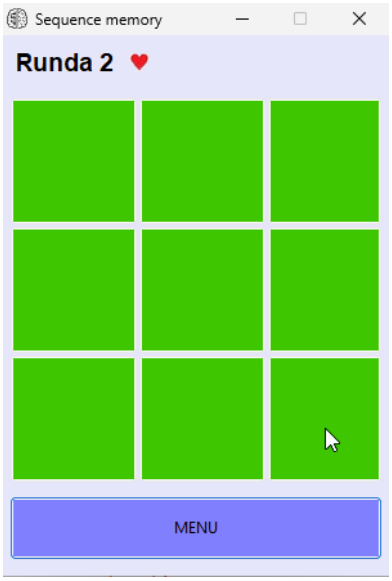
(a) Poziom atwy (b) Poziom redni (c) Poziom trudny

Rysunek 2: Rónorodno poziomów

Efekty Nacinicia Poprawnego Kwadratu lub Przegrania

Po kadym naciniciu kwadratu na planszy, gra reaguje zgodnie z poniszymi zasadami:

- **Nacinicie Poprawnego Kwadratu:** Jeli gracz poprawnie nacinie kwadrat zgodnie z aktualn sekwencj, gra rejestruje poprawne nacinicie, a plansza wywietla si na zielony kolor i przechodzi do kolejnego etapu sekwencji (rysunek 3).



Rysunek 3: Uzyskanie poprawnej odpowiedzi

- **Przegrana:** Jeli gracz popeni bd i nacinie niewaciwy kwadrat, liczba y zmniejsza si, a pola podwietlaj si kolor czerwony.

  Jeli liczba y osignie zero, gra koczy si. Na ekranie pojawia si inforamcja o przegranej oraz rundzie na której zakoczya si rozgrywka. (rysunek 4).

Warto zauway, e wybór poziomu trudnoci wpywa nie tylko na sam rozgrywk, ale take na dowiadczenie uytkownika, dostosowujc si do rónych poziomów umiejtnoci i preferencji.

Rysunek 4: Uzyskanie zej odpowiedzi

Struktura Kodu

Kod ródowy skada si z dwóch plików nagówkowych oraz dwóch plików cpp.

seqMain.h

Plik nagówkowy w jzyku C++ definiuje gówn klas dialogow (seqDialog) dla aplikacji opartej na bibliotece wxWidgets. Oto kilka kluczowych elementów pliku:

- **Doczenie bibliotek:** Plik docza standardowe nagówki biblioteki C++ (<vector> i <map>) oraz zewntrzny plik nagówkowy "seq.h".

```
//...
#include <vector>
#include "seq.h"
#include <map>
//...
```

Listing 1:

- **Deklaracja klasy:** Jest to deklaracja klasy seqDialog, dziedziczcej po klasie wxDialog, która reprezentuje gówne okno dialogowe aplikacji.

```
class seqDialog: public wxDialog
{
    // ... (tre klasy)
};
```

Listing 2:

- **Metody publiczne:** Na pocztku deklarowany jest konstuktor i destruktor klasy seqDialog. Nastpnie dodalimy wasne funckje umoliwiajce lepsz przejysto kodu i zmniejszenie redundancji.

```
//...
public:
    seqDialog(wxWindow* parent,wxWindowID id = -1);
    virtual ~seqDialog();

    void RepeatUserSequence();
    void StanPoczatkowyPola(std::vector<wxButton*> pola,
       wxColour kolor, bool wlacz);
    void CzyMenu(bool tak);
    void UstawRozmiar(int szerokosc, int wysokosc);
//...
```

Listing 3:

- ***void RepeatUserSequence();***Metoda, która odpowiada za powtórzenie sekwencji, w przypadku gdy uytkownik si pomyli i ma jeszcze ycia.
- ***void StanPoczatkowyPola(std::vector<wxButton*> pola, wxColour kolor, bool wlacz);*** Metoda, która zmienia pola (kwadraty). Moe zmieni kolor ta przycisków a take ustawi czy s wczone, czy wyczone (czy mona w nie klikn).
- ***void CzyMenu(bool tak);*** Metoda odpowiedzialna za wywietlenie bd schowanie interfejsu menu w odpowiednich sytuacjach.
- ***void UstawRozmiar(int szerokosc, int wysokosc);*** Metoda ustawiajca rozmiar okna dialogowego gry, w zalenosci od poziomu trudnoci.

- **Pola prywatne:** Stworzylimy kilka zmiennych, uywanych przez nasz gre.

```
//...
wxFlexGridSizer* FlexGridSizer2;
std::vector<wxButton*> pola;
int liczba_pol;
std::map<int,int> id2nr;
wxBitmap serce;
wxColour kolory[4];
Sequence * gra;
vector<int> userSeq;
int liczba_rund = 1;
wxString w;
int zycia;
int poziom_trudnosci;
wxFlexGridSizer* sizer;
wxColour backgroundColor;
bool sizer_ustawiony=false;
//...
```

Listing 4:

- ***wxFlexGridSizer* FlexGridSizer2;*** Jest to wskanik na obiekt typu wxFlexGridSizer, który reprezentuje siatk do uoenia elementów w interfejsie uytkownika.
- ***std::vector<wxButton*> pola;*** Jest to wektor przechowujcy wskaniki do obiektów typu wxButton. Suy do przechowywania przycisków (kwadratów) w interfejsie uytkownika.
- ***int liczba_pol;*** Zmienna przechowujca liczb pól. Uywana jest do utworzenia rónych rozmiarów planszy w zalenoci od poziomu trudnoci.
- ***std::map<int,int> id2nr;*** Jest to mapa (asocjacyjna tablica), która mapuje identyfikatory przycisków wxButton (int) na odpowiadajce im numery (int) w wektorze.
- ***wxBitmap serce;*** Jest to obiekt reprezentujcy bitmap (obraz), która reprezentuje grafik serca. Uywana w celu wywietlenia serc w zalenoci od liczby y.
- ***wxColour kolory[4];*** Tablica przechowujca cztery obiekty typu wxColour, reprezentujce róne kolory. Uywana w kontekcie kolorowania elementów interfejsu uytkownika.
- ***Sequence * gra;*** Wskanik na nowy obiekt typu Sequence, co sugeruje, e klasa seqDialog korzysta z klasy Sequence.
- ***vector<int> userSeq;*** Wektor przechowujcy sekwencj liczb cakowitych, któr wprowadzi uytkownik, poprzez kliknicie w odpowiednie pola.
- ***int liczba_rund = 1;*** Zmienna przechowujca liczb rund. Suy do ledzenia postpu gry.
- ***wxString w;*** Obiekt reprezentujcy cig znaków (wxString), suy do przechowywania komunikatów dla uytkownika.
- ***int zycia;*** Zmienna przechowujca liczb y w grze. Uywana w kontekcie gry, gdzie gracz ma okrelon liczb szans. Warto zostanie pobrana z przycisku wxChoice.

- ***int poziom_trudnosci;*** Zmienna przechowujca poziom trudnoci gry. Im wikszy poziom trudnoci, tym wicej pól pojawia si na planszy. Warto zostanie pobrana z przycisku wxChoice.
- ***wxFlexGridSizer* sizer;*** Wskanik na obiekt typu wxFlexGridSizer. Jest uywany do ustawiania pól w odpowiednim ukadzie.
- ***wxColour backgroundColor;*** Obiekt reprezentujcy kolor ta w interfejsie uytkownika.
- ***bool sizer_ustawiony=false;*** Zmienna typu boolowskiego, która informuje, czy sizer zosta ju ustawiony. Uywana do sprawdzania, czy ukad interfejsu zosta ju skonfigurowany.

seq.h

Plik nagówkowy w jzyku C++ definiuje klase seq dla aplikacji opartej na bibliotece wxWidgets. Oto kilka kluczowych elementów pliku:

- **Doczenie bibliotek:** Plik docza standardowe nagówki biblioteki C++ (<iostream> i <vector>) oraz definiuje przestrze nazw jako std.

```
//...
#include <iostream>
#include <vector>

using namespace std;
//...
```

Listing 5:

- **Deklaracja klasy:** Jest to deklaracja klasy Sequence, która reprezentuje logik gry.

```
class Sequence
{
    //...
};
```

Listing 6:

- **Deklaracja klasy:** Klasa Sequence zawiera klika zmiennych oraz metod publicznych.

```
//...
public:
vector<int> seq;
int rozmiar_kwadratu=3;
int liczba_zyc=1;
Sequence();
~Sequence();
void addElement();
int get_rozmiar_kwadratu();
int get_liczba_zyc();
void set_rozmiar_kwadratu(int rozmiar);
void set_liczba_zyc(int liczba);
//...
```

Listing 7:

  - ***vector<int> seq;*** Wektor przechowujcy sekwencj liczb cakowitych, która jest generowana automatycznie.
  - ***int rozmiar-kwadratu=3;*** Zmienna przechowujca domyln ilo kwadratów w pionie i poziomie.
  - ***int liczba-zyc=1;*** Zmienna przechowujca domyln liczb y w grze.
  - Nastpnie jest deklaracja konstruktora i destruktora klasy.
  - ***void addElement();*** Metoda dodajca element do sekwencji.
  - ***int get-rozmiar-kwadratu();*** Metoda zwracajca ilo kwadratów w pionie i poziomie.
  - ***int get-liczba-zyc();*** Metoda zwracajca liczb y.
  - ***void set-rozmiar-kwadratu(int rozmiar);*** Metoda ustawiajca liczb kwadratów w pionie i poziomie na argument rozmiar.
  - ***void set-liczba-zyc(int liczba);*** Metoda ustawiajca liczb y na argument liczba.

seq.cpp

Plik zawiera implementacj klasy Sequence. Oto jego podstawowe elementy:

- **Pocztek pliku:** Na poczrtku doczony jest plik "seq.h", ustawiona przestrze nazw na std oraz zaimplementowany konstruktor i destruktor.

```
#include "seq.h"
using namespace std;
Sequence::Sequence(){}
Sequence::~Sequence(){}
```

Listing 8:

- **addElement** Procedura add element dodaje element do vektora seq. Element jest generowany automatycznie to liczba losowa z przedziau od 0 do iloci pól na planszy.

```
void Sequence::addElement(){
    int losowaLiczba = std::rand() %
       (rozmiar_kwadratu*rozmiar_kwadratu);
    seq.push_back(losowaLiczba);
}
```

Listing 9:

- **get_rozmiar_kwadratu** Metoda get rozmiar kwadratu zwraca liczb cakowit przechowywan w zmiennej rozmiar kwadratu.

```
int Sequence::get_rozmiar_kwadratu(){
    return rozmiar_kwadratu;
}
```

Listing 10:

- **get_liczba_zyc** Metoda get liczba zyc zwraca liczb cakowit przechowuwan w zmiennej liczba y.

```
int Sequence::get_liczba_zyc(){
    return liczba_zyc;
}
```

Listing 11:

- **set_rozmiar_kwadratu** Procedura ustawia rozmiar kwadratu obiektu klasy sequence na argument rozmiar.

```
void Sequence::set_rozmiar_kwadratu(int rozmiar){
    rozmiar_kwadratu = rozmiar;
}
```

Listing 12:

- **set_liczba_zyc** Procedura ustawia liczb y obiektu klasy sequence na argument liczba.

```
void Sequence::set_liczba_zyc(int liczba){
    liczba_zyc = liczba;
}
```

Listing 13:

seqMain.cpp

Plik implementuje okno dialogowe aplikacji. Oto przykadowe funkcjonalnoci w tym pliku:

- **Pocztek pliku:** Na pocztku pliku oprócz standardowych akcji wykonywanych przez wxWidgets dodalimy funkcj, która umoliwia wywietlanie znaków polskich w aplikacji.

```
#undef _
#define _(s) wxString::FromUTF8(s)
```

Listing 14:

- **konstruktor klasy seqDialog:** Ustawia on odpowiednie pola wstawione przez wxWidgets. Dodalimy do niego klika funkcjonalnoci.

```
seqDialog::seqDialog(wxWindow* parent,wxWindowID id)
{
    //...
    Button2->Show(false);
    serce = wxBitmap(wxImage("serce.png"));
    kolory[0] = wxColour(200, 150, 255);
    kolory[1] = wxColour(171, 32, 253);
    kolory[2] = wxColour(255,0,0);
    kolory[3] = wxColour(0,200,0);
    backgroundColor = wxColour(230, 230, 250);
    sizer = new wxFlexGridSizer(3, 3, 0, 0);
    gra = nullptr;
    srand(time(0));
    UstawRozmiar(240,240);
    SetBackgroundColour(backgroundColor);
    this->FlexGridSizer2 = FlexGridSizer2;
    SetIcon(wxICON(aaaa));
}
```

Listing 15:

  - ***Button2->Show(false);*** Wyczenie widocznosci przycisku 2 (odpowiada on za powrót do menu).
  - ***serce = wxBitmap(wxImage("serce.png"));*** Przypisanie zmiennej serce odpowiadajcej pliku.
  - ***kolory[i]=...*** Przypisanie kolorów uywanych w aplikacji.
  - ***backgroundColor = wxColour(230, 230, 250);*** Przypisanie koloru ta okna dialogowego.
  - ***sizer = new wxFlexGridSizer(3, 3, 0, 0);*** Utworzenie nowego sizera, przechowujcego pola do gry.
  - ***gra = nullptr;*** Przypisanie wkanikowi gry pustego obiektu.
  - ***srand(time(0));*** Inicjalizuje generator liczb pseudolosowych przy uyciu aktualnego czasu.
  - ***UstawRozmiar(240,240);*** Wywouje funkcj ustaw rozmiar, która zmienia rozmiar okna dialogowego na 240x240px.
  - ***SetBackgroundColour(backgroundColor);*** Zmienia kolor ta okna dialogowego na ustawiony wczeniej kolor.
  - ***this->FlexGridSizer2 = FlexGridSizer2;*** Przypisuje wskanik FlexGridSizer2 do zmiennej FlexGridSizer2 w klasie seqDialog.
  - ***SetIcon(wxICON(aaaa));*** Ustawia ikon okna dialogowego.

- **PrzyciskStart:** Zwizany jest ze startem gry, po nacisniciu przycisu start i wybraniu odpowiednich opcji. Oto jego funkcje:
  - Pobranie poziomu trudnoci z przycisku Choice1 i liczby y z przyciusku Choice2.

```
//...
poziom_trudnosci = Choice1->GetCurrentSelection();
zycia = Choice2->GetCurrentSelection()+1;
//...
```

Listing 16:

– Przygotowanie nowej gry i usunicie starej w przypadku, gdy nie zostaa jeszcze usunita

```
//...
if (gra != nullptr){
    delete gra;
    gra = nullptr;
}
gra = new Sequence();
//...
```

Listing 17:

– Dodanie do wczeniej przygotowanej gry nowego elementu, ustawienie liczby y na warto pobran z pola Choice2 i ustawienie rozmiaru kwadratu (liczby wywietlanych pól) w zalenoci od poziomu trudnoci.

```
//...
gra->addElement();
gra->set_rozmiar_kwadratu(poziom_trudnosci+3);
gra->set_liczba_zyc(zycia);
//...
```

Listing 18:

– Ukrycie menu, poprzez wywoanie funkcji czy menu z argumentem false.

```
//...
CzyMenu(false);
//...
```

Listing 19:

– Ustawienie liczby kolumny i werszy sizerowi, który przechowuje pola do gry. (liczba wierszy i kolumn s równe i zale od poziomu trudnoci wybranego przez uytkownika).

```
//...
sizer->SetCols(gra->get_rozmiar_kwadratu());
sizer->SetRows(gra->get_rozmiar_kwadratu());
//...
```

Listing 20:

– Utworzenie nowych przycisków, które s polami do gry. Tworzymy je w ptli od 0 do liczby kolumn razy liczba wierszy. Ustawiamy tym polom rozmiar 100x100px. Dodajemy do sizera te pola oraz czymy je z funkcj PrzyciskPola, wywoujc si po klikniciu danego przycisku. Na kocu dodajemy utworzone pola do wektora przechowujcego te pola.

```
//...
for (int i = 0; i < gra->get_rozmiar_kwadratu() *
    gra->get_rozmiar_kwadratu(); i++)
{
    wxButton* nowe_pole = new wxButton(this,
        wxID_ANY, wxEmptyString, wxDefaultPosition,
        wxDefaultSize, 0, wxDefaultValidator,
        wxString::Format("ID_BUTTON%d", i));
    nowe_pole->SetSize(wxSize(100, 100));
    nowe_pole->Disable();
    sizer->Add(nowe_pole, 1, wxEXPAND |
        wxFIXED_MINSIZE, 5);
    Connect(nowe_pole->GetId(),
        wxEVT_COMMAND_BUTTON_CLICKED,
        (wxObjectEventFunction)&seqDialog::PrzyciskPole);

    pola.push_back(nowe_pole);
}
//...
```

Listing 21:

– Przypisanie id przycisków z wektora pola odpowiadajcym im numerom indeksów w wektorze.

```
//...
for(int i = 0;
    i<gra->get_rozmiar_kwadratu()*gra->get_rozmiar_kwadratu();
    i++){
     id2nr[pola[i]->GetId()]=i;
}
//...
```

Listing 22:

– Ustawienie pól na kolor podstawowy, wyczenie moliwoci ich kliknicia.

```
//...
StanPoczatkowyPola(pola, kolory[0], false);
//...
```

Listing 23:

– Dodanie sizera przechowujcego pola do gry do FlexGridSizer2 tylko, gdy jeszcze nie by dodany.

```
//...
if (!sizer_ustawiony){
     FlexGridSizer2->Add(sizer, 1, wxEXPAND);
     sizer_ustawiony= true;
}
//...
```

Listing 24:

– Przeliczenie szerokoci i wysokoci okna dialogowego w zalenoci od liczby pól.

```
//...
int szerokosc = gra->get_rozmiar_kwadratu() * 100;
int wysokosc = gra->get_rozmiar_kwadratu() * 100 +
    120;
//...
```

Listing 25:

– Wyczyszczenie komunikatu uytkownika. Przypisanie do niego liczby rund. Zmiana Tekstu w StaticText1 na liczb rund.

```
//...
w.Clear();
liczba_rund = 1;
w<<"Runda␣";
w<<liczba_rund;
StaticText1->SetLabel(w);
//...
```

Listing 26:

– Wywietlenie odpowiedniej liczby serc na ekranie.

```
//...
if(zycia>=1){
     StaticBitmap1->SetBitmap(serce);
}
if(zycia>=2){
     StaticBitmap2->SetBitmap(serce);
}
if(zycia>=3){
     StaticBitmap3->SetBitmap(serce);
}
```

```
11                  //...
```

Listing 27:

– Wyczyszczenie sekwencji uytkownika oraz komunikatu do uytkownika.

```
1                   //...
2                   userSeq.clear();
3                   w.Clear();
4                   //...
```

Listing 28:

– Podwietlenie wybranego przycisku na 500 milisekund i wywoanie funkcji onTimer.

```
1                   //...
2                   StanPoczatkowyPola(pola, kolory[1], false);
3
4                   wxTimer* timer = new wxTimer(this, wxID_ANY);
5                   timer->StartOnce(500);
6                   Connect(timer->GetId(), wxEVT_TIMER,
                        wxTimerEventHandler(seqDialog::OnTimer));
7                   //...
```

Listing 29:

– Zmiana rozmiaru okna dialogowego.

```
1                   //...
2                   UstawRozmiar(szerokosc, wysokosc);
3                   //...
```

Listing 30:

- **PrzyciskMenu:** Kod funkcji PrzyciskMenu jest zwizany z powrotem do menu gównego gry. Oto opis kluczowych kroków w tej funkcji:

– Zwolenienie pamici zajmowanej przez obiekt gra.

```
1                   //...
2                   if (gra != nullptr){
3                       delete gra;
4                       gra = nullptr;
5                   }
6                   //...
```

Listing 31:

– Usuwanie przycisków pola, wyczyszczenie wektora pola oraz sizera przechowujcego te pola.

```
1                   //...
2                   for(auto pole : pola){
3                       pole->Destroy();
4                   }
5                   pola.clear();
6                   sizer->Clear();
7                   //...
```

Listing 32:

– Pokazanie przycisków menu, usunicie widoku serc.

```
1                   //...
2                   CzyMenu(true);
3                   StaticBitmap1->SetBitmap(wxNullBitmap);
4                   StaticBitmap2->SetBitmap(wxNullBitmap);
```

```
5                   StaticBitmap3->SetBitmap(wxNullBitmap);
6                   //...
```

Listing 33:

– Zmiana komunikatu uytkownika.

```
1                   //...
2                   w.clear();
3                   w << "WYBIERZ USTAWIENIA";
4                   StaticText1->SetLabel(w);
5                   //...
```

Listing 34:

– Zmiana rozmiaru okna dialogowego.

```
1                   //...
2                   UstawRozmiar(240,240);
3                   //...
```

Listing 35:

- **PrzyciskPole:** Funkcja PrzyciskPole obsuguje zdarzenie nacinicia przycisku na planszy gry. Oto kluczowe kroki tej funkcji:

  – Pobranie id pola, które zostao kliknite. Zamiana tego ID na numer pola w wektorze pola. Dodanie tego numeru do wektora userSeq.

```
1                   //...
2                   int numer_pola = id2nr[event.GetId()];
3                   userSeq.push_back(numer_pola);
4                   //...
```

Listing 36:

– Sprawdzenie sekwencji uytkownika

```
1                   //...
2                   if (userSeq.size() <= gra->seq.size()) {
3                       int index = userSeq.size() - 1;
4                       if (userSeq[index] != gra->seq[index]) {
5                           gra->set_liczba_zyc(zycia--);
6                           StanPoczatkowyPola(pola, kolory[2], false);
7                           if (zycia > 0) {
8                               if(zycia==2){
9                                   StaticBitmap1->SetBitmap(wxNullBitmap);
10                              }
11                              if (zycia==1){
12                                  StaticBitmap2->SetBitmap(wxNullBitmap);
13                              }
14                              Sleep(500);
15                              RepeatUserSequence();
16                          } else {
17                              StaticBitmap3->SetBitmap(wxNullBitmap);
18                              w.Clear();
19                              w << L"Przegrae Osignito limit bdów. 
                                    \nZakoczye na rundzie " <<
                                    liczba_rund;
20                              wxMessageBox(w, wxString("Koniec gry"));
21                          }
22                      } else if (userSeq.size() == gra->seq.size()) {
23                          gra->addElement();
24
```

```
                        w.Clear();
                        liczba_rund++;
                        w << "Runda " << liczba_rund;
                        StaticText1->SetLabel(w);

                        wxTimer* timer = new wxTimer(this, wxID_ANY);
                        Connect(timer->GetId(), wxEVT_TIMER,
                            wxTimerEventHandler(seqDialog::OnTimer));
                        timer->SetClientData(reinterpret_cast<void*>(0));
                            // Zainicjowanie indeksu
                        timer->Start(500, wxTIMER_ONE_SHOT);

                        userSeq.clear();
                        StanPoczatkowyPola(pola, kolory[3], false);
                    }
                }
                //...
```

Listing 37:

Warunek `userSeq.size() <= gra->seq.size()` sprawdza, czy dugo sekwencji wprowadzonej przez uytkownika jest mniejsza lub równa dugoci sekwencji generowanej przez gr. To jest sprawdzane, aby upewni si, e uytkownik nie przekroczy jeszcze dugoci sekwencji generowanej przez gr.

Jeli wystpi bd (czyli aktualne pole uytkownika nie zgadza si z oczekiwanym polem w sekwencji), zostaje wykonany blok kodu wewntrz tego warunku. Wtedy:

* Liczba y (zycia) zostaje zaktualizowana.
* Kolory pól s zmieniane na czerwony.
* Sprawdzane jest, czy uytkownik ma jeszcze ycia. Jeli tak, aktualizuje si interfejs i powtarza si sekwencj. W przeciwnym razie, gra jest zakoczona, a uytkownikowi wywietla si komunikat o przegranej.

Jeli dugo sekwencji wprowadzonej przez uytkownika jest równa dugoci sekwencji generowanej przez gr, to oznacza poprawn sekwencj. Wtedy:

* Dodawany jest nowy element do sekwencji generowanej przez gr.
* Zmieniana jest liczba rund.
* Uruchamiany jest timer do podwietlania sekwencji.
* Sekwencja wprowadzona przez uytkownika jest czyciszona, a kolory pól zmieniane s na zielony.

- **RepeatUserSequence:** Funkcja RepeatUserSequence suy do ponownego podwietlenia sekwencji, na której zakoczy uytkownik w przypadku pomyki.
  - Przywrócenie stanu pocztkowego pól.

```
                //...
                StanPoczatkowyPola(pola, kolory[0], false);
                //...
```

Listing 38:

  - Ponowne podwietlenie sekwencji na której uytkownik si pomyli.

```
                //...
                wxTimer* timer = new wxTimer(this, wxID_ANY);
                Connect(timer->GetId(), wxEVT_TIMER,
                    wxTimerEventHandler(seqDialog::OnTimer));
                timer->SetClientData(reinterpret_cast<void*>(0));
                timer->Start(500, wxTIMER_ONE_SHOT);
                //...
```

Listing 39:

  - Wyczyszczenie sekwencji wprowadzonej przez uytkownika.

```
//...
userSeq.clear();
//...
```

Listing 40:

- **OnTimer:** Funkcja OnTimer obsuguje zdarzenia timera. Wywoywana jest po upywie okrelonego czasu i odpowiedzialna jest za animowanie sekwencji.
  - Pobiera indeks z informacji klienta przekazanej przez timer.

```
//...
size_t index =
    reinterpret_cast<size_t>(event.GetTimer().GetClientData());
//...
```

Listing 41:

Jeli obiekt gra istnieje to:

  - Przywraca stan pocztkowy pól.

```
//...
StanPoczatkowyPola(pola, kolory[0], false);
//...
```

Listing 42:

Jeli index pola kliknitego jest mniejszy od rozmiaru sekwencji gry to:

  - Przywró stan pocztkowy poprzedniego przycisku, zanim podwietli si kolejny.

```
//...
int x = gra->seq[index];
int poprzedni = gra->seq[index - 1];
pola[poprzedni]->SetBackgroundColour(kolory[0]);
pola[poprzedni]->Refresh();
pola[x]->Update();
Sleep(500);
//...
```

Listing 43:

  - Podwietl kolejne pole.

```
//...
pola[x]->SetBackgroundColour(kolory[1]);
pola[x]->Refresh();
//...
```

Listing 44:

  - Uruchom timer ponownie, przeka dane o indeksie o 1 wikszym ni wczeniej.

```
//...
wxTimer* timer = new wxTimer(this, wxID_ANY);
Connect(timer->GetId(), wxEVT_TIMER,
    wxTimerEventHandler(seqDialog::OnTimer));
timer->SetClientData(reinterpret_cast<void*>(index +
    1));
timer->Start(500, wxTIMER_ONE_SHOT);
//...
```

Listing 45:

Jeli index jest wiekszy od rozmiaru sekwencji w obiekcie gry, to:

– Przywró stan pocztkowy pól, wcz pola.

```
                //...
            StanPoczatkowyPola(pola, kolory[0], true);
                //...
```

Listing 46:

- **PrzyciskZasady:** Przycisk zasady wywietla zasady gry w MessageBoxie.
- **StanPocztkowyPola:** okrela kolory pól w wektorze pola oraz wcza lub wycza moliwo kliknicia w te pola.

```
            void seqDialog::StanPoczatkowyPola(std::vector<wxButton*>
                pola, wxColour kolor, bool wlacz){
                for (auto pole : pola) {
                    pole->SetBackgroundColour(kolor);
                    pole->Enable(wlacz);
                }
            }
```

Listing 47:

- **CzyMenu:** pokazuje lub chowa menu w zalenoci od argumentu.

```
            void seqDialog::CzyMenu(bool tak){
                StaticText2->Show(tak);
                StaticText3->Show(tak);
                StaticText4->Show(tak);
                Choice1->Show(tak);
                Choice2->Show(tak);
                Button1->Show(tak);
                Button3->Show(tak);
                Button2->Show(!tak);
            }
```

Listing 48:

- **UstawRozmiar** ustawia rozmiar okna dialogowego na podane argumenty.

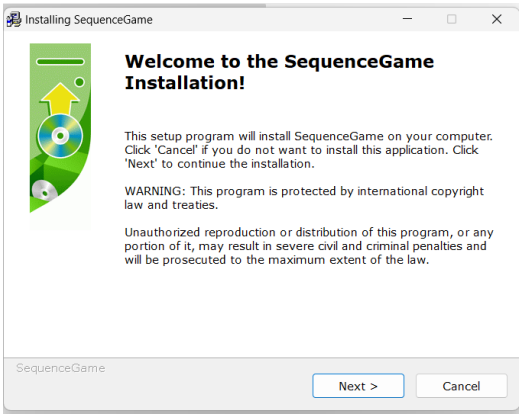
```
            void seqDialog::UstawRozmiar(int szerokosc, int wysokosc){
                SetClientSize(wxSize(szerokosc, wysokosc));
                Layout();
            }
```

Listing 49:

Instrukcja instalacji

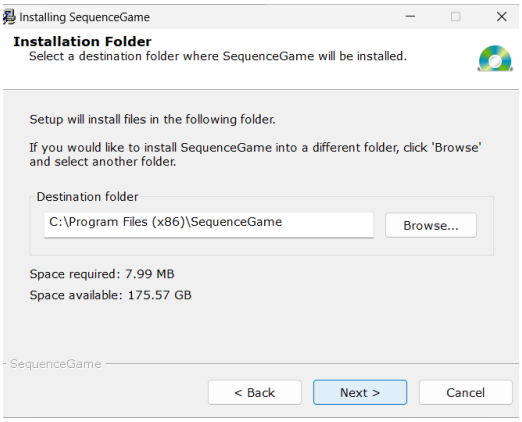
Instalator gry powsta przy pomocy aplikacji Createinstall. Aby zainstalowa gr naley postpowa wedug poniszych kroków.

1. Uruchom pakiet instalacyjny.
2. Zapoznaj si z wywietlonymi informacjami, a nastpnie przejd dalej, naciskajc przycisk „Next”.



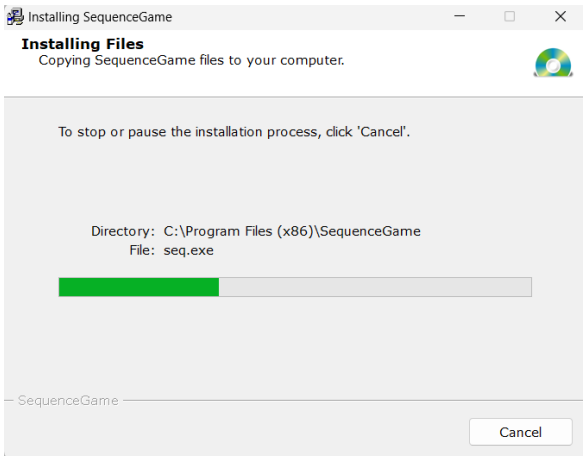
Rysunek 5: Wybór lokalizacji folderu.

3. Wybierz lokalizacj folderu, a nastpnie przejd dalej.

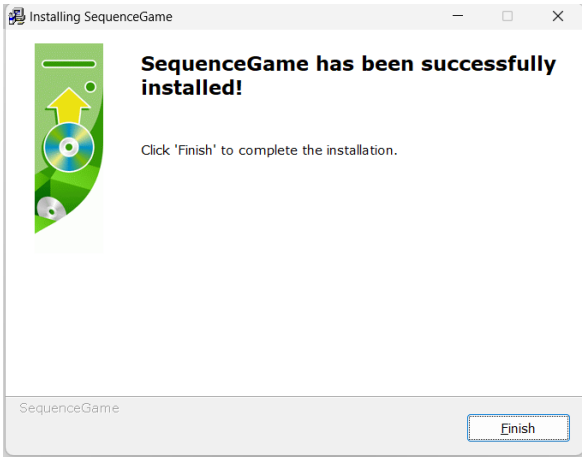


Rysunek 6: Wybór lokalizacji folderu.

4. Poczekaj, a aplikacja zostanie zainstalowana. W kadej chwili moesz przerwa ten proces, uywajc przycisku „Canel”.
5. Na zakoczenie pojawi si okno dialogowe, informujce o pomylnym zakoczeniu aplikacji. Aby zakoczy cay proces nacinij „Finish” .
6. Uruchom gr i ciesz si rozgrywk.



(a) Instalacja



(b) Zakoczenie instalacji

Podsumowanie

Gra Sequence Memory wyrónia si estetycznymi elementami wizualnymi, jal równie dynamicznym i intuicyjnym interfejsem graficznym, co uatwia uytkowikom szybkie przyswajanie zasad. Generowanie rónych rodzajów sekwencji na trzech poziomach trudnoci zapewnia graczom nowe wyzwania.

„Sequence Memory” to nie tylko gra, ale take narzdzie rozwijajce umiejtnoci kognitywne graczy w atrakcyjny sposób. Wzbogacona o elementy edukacyjne, stanowi idealn propozycj dla tych, którzy chc jednoczenie bawi si i rozwija wasne zdolnoci pamiciowe.Warto wspomnie, e nie ma okrelonego przedziau wiekowego osób, mogcych korzysta z aplikacji. Kady znajdzie bowiem odpowiednie ustawienia dla siebie.