System wizualizacji warcab

Dokumentacja

Jan Augustyniak
Dominik Garczyk
Sebastian Szeszko

8 czerwca 2017

Spis treści

1 Opis aplikacji

1.1 Cele aplikacji

Zadaniem projektu jest wizualizacja obrazu gry w warcaby na komputerze. Obraz gry pobierany jest z kamery umieszczonej nad planszą. Odpowiedni podsystem wyszukuje pozycje pionków i przekazuje je do podsystemu wizualizacji. Pozycje pionków są odwzorowywane w macierzy programu. Dodatkowo, aplikacja ma sprawdzać poprawność wykonanych ruchów.

1.2 Uzasadnienie wyboru

Grupa wybrała ten temat, ponieważ wcześniej nie wykorzystywała w projektach urządzeń odbierających dane ze świata zewnętrznego (kamera). Ponadto, przetwarzanie obrazu to obecnie jedno z popularnych zastosowań współczesnych komputerów.

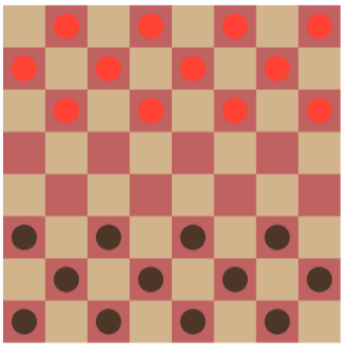
1.3 Założenia projektowe

Plansza do gry powinna posiadać 64 kwadratowe pola (8x8) oraz powinna być wokół obramowana. Bok kwadratowego pola powinnien być co najmniej 0,5 cm większy od średnicy pionka. Zalecane, aby kolory pionków obu graczy posiadały jak największy kontrast, w celu łatwiejszego zidentyfikowania przynależności pionków do konkretnego gracza. Urządzenie rejestrujące obraz musi znajdować się nad planszą, najlepiej w miejscu środka planszy. Załozenie jest takie, że rejestrowana plansza musi być położona prosto, bez wyraźnych odchyleń. Dla testowanej szachownicy 28x28 cm, należało umiejscowić kamerę 50 cm nad planszą. W celu zapobiegnięcia przekłamań podczas odczytu danych z obrazu, należy równomiernie i dostatecznie jasno oświetlić obiekty gry.

Gra odbywa się na zasadach warcab, w wersji angielskiej:

- Plansza 8x8

- 12 pionów dla każdego z graczy
- Piony przesuwają się po skosie o jedno pole jeżeli nie jest ono zajęte
- Jeżeli to pole możliwego ruchu jest zajęte przez pion przeciwnika, a bezpośrednio za tym pionem, na tej samej przekątnej, jest puste pole, można wtedy zbić ten pion przeciwnika, przez postawienie swojego piona na tym pustym polu
- Jeżeli gracz ma możliwość w swoim ruchu dokonania bicia, gracz musi to zrobić
- Jeżeli po biciu gracz ma możliwość kolejnego bicia, gracz musi je wykonać w tym samym posunięciu, jest to tak zwane bicie wielokrotne
- Gdy jest więcej niż jedna możliwa droga bicia, możemy ją dowolnie wybrać
- Pion dochodzący do ostatniego rzędu staje się królem
- Koronacja pionka powoduje zakończenie posunięcia (nie można koronować pionka przez zbicie, a później zbić nim kolejnego pionka bijąc do tyłu)
- Król ma analogiczną możliwość wykonywania posunięć jak pion, z tym, że może poruszać się i bić także do tyłu
- Wygrana następuje po zbiciu wszystkich pionków przeciwnika, lub uniemożliwieniu wykonania przez nie ruchu
- Remis następuje po wykonianiu 50 ruchów bez żadnego bicia



Rysunek 1: Poprawne ustawienie początkowe

1.4 Działanie aplikacji

1.4.1 Ustawienie źródła obrazu

Po włączeniu aplikacji użytkownik musi podać źródło wideo. Z listy rozwijanej ma on możliwość wybrania połączenia się z telefonem i przekazu obrazu z kamery za pomocą aplikacji DroidCam.

1.4.2 Wykrywanie kolorów

Po rozpoczęciu transmisji obrazu w oknie po prawej częsci aplikacji pojawia się podgląd transmisji. W każdej klatce obrazu wyszukiwany jest kwadrat a następnie jest on dzielony na 64 równe częsci odzwierciedlające pola na szachownicy. Jeśli kwadrat zostanie wykryty aplikacja rozpoczyna wyszukiwanie pionków. Krawędzie wykrytych pionków są kolorowane w celu poinformowania użytkownika o ich wykryciu. Gdy użytkownik zaznaczy opcję "Wykryj kolory pionków", aplikacja zapisze kolory wykrytych kół na obrazie oraz wyświetli je w okienkach. Użytkownikowi przedstawione zostaną rónież ich wartości RGB. Po wykryciu kolorów pionków obu graczy użytkownik odznacza pole "Wykryj kolory pionków". Użytkownik ma możliwość dostosowania zakresów kolorów pionków graczy za pomocą suwa-

ków umieszczonych pod oknami wyświetlającymi kolory.

1.4.3 Wykrywanie pionków

Gdy pole "Wykryj kolory automatycznie"jest odznaczone aplikacja dwa razy filtruje każdą klatkę otrzymanego wideo pozostawiając tylko te elementy, które mają kolory zapisane jako kolory pionków graczy (raz pozostawia tylko kolory pionków gracza pierwszego, drugi raz pozostawia tylko kolory pionków gracza drugiego). Podgląd wideo po filtrowaniu jest udostępniany użytkownikowi w dolnej częsci aplikacji osobno dla pierwszego jak i dla drugiego gracza. Wykryte pionki wyświetlane są na planszy w głównej częsci aplikacji. Pionki pierwszego z graczy wyświetlane są jako czarne pola natomiast pionki drugiego z graczy wyświetlane są jako białe pola.

1.4.4 Wykonywanie ruchów

Gdy Użytkownik wykona ruch pionkiem klika on przycisk "Wykonaj ruch". Po jego kliknięciu aplikacja sprawdza poprawność ruchu. Jeśli nie został on wykonany poprawnie, użytkonwik jest o tym fakcie informowany a ruch nie jest zapisywany. Jeśli został wykonany poprawnie, ruch zapisywany jest w historii ruchów. Po wykonaniu pierwszego przesunięcia i zapisaniu go użytkownik w każdej chwili ma możliwość wyświetlenia stanu planszy po każdym z już wykonanych ruchów.

2 Funkcjonalności

2.1 Transmisja obrazu, wyświetlanie

2.1.1 Wykrywanie kolorów

- Aplikacja umożliwia automatyczne wykrycie kolorów pionków na szachownicy.
- Ręczne dostosowywanie zakresu kolorów pionków graczy
- Filtrowanie obrazu
- Aplikacja filtruje każdą klatkę otrzymywanego wideo pozostawiając jedynie kolory pionków graczy.

2.2 Sprawdzanie poprawności wykonania ruchu

Po kliknięciu przycisku "Wykonaj ruch"przez użytkownika aplikacja sprawdza, czy ruch został wykonany prawidłowo. W tym celu sprawdzamy wszystkie możliwości wykoniania prawidłowego ruchu przez danego gracza i sprawdzamy czy ruch, który ten gracz wykonał znajduje się na tej liście.

2.3 Historia wykonanych ruchów

Po wykonaniu każdego poprawnego ruchu stan planszy jest zapisywany przez aplikację. Użytkownik po wykonaniu pierwszego ruchu w każdym momencie jest w stanie wyświetlić wszystkie ruchy wykonane od początku gry.

3 Technologie

Grupa projektowa musiała dokonać wyboru odpowiednich narzędzi w celu przystępnego zrealizowania modułów takich jak: rejestracja przebiegu gry, analiza obrazu, wykrywanie planszy oraz pionków czy też weryfikacja ruchów. Do utworzenia projektu wybrano środowisko programistyczne Visual Studio, ze względu na znajomość programu przez wszystkich członków grupy oraz fakt, iż nie są wymagane dodatkowe aplikacje umożliwiające funkcjonowanie pisanego programu. Dodatkowo, IDE produkcji Microsoft'u posiada system zarządzania pakietami NuGet, przez co zbyteczne jest pobieranie bibliotek EmguCV z zewnętrznych źródeł - odpowiednie odwołania zostaną dodane do projektu. Jako język programowania użyto obiektowego C, który był wcześniej wykorzystywany przez członków grupy. Za silnik graficzny posłużył WPF dostępny w Visual Studio. Zarządzanie kodem odbywało się w ramach publicznego repozytorium dodanego na platformie GitHub.

3.1 C#

C# – obiektowy język programowania. Jest zgodny z .NET Framework. Cieszy się dużą popularnością, posiada liczne biblioteki. Charakteryzuje sie tym, że zarządzaniem pamięcią zajmuje się środowisko uruchomieniowe. W rezultacie, programista nie musi pamietać o zwalnianiu pamięci po nieużywanych obiektach.

3.2 EmguCV

EmguCV (wykorzystywana wersja: 3.1.0) to darmowa biblioteka funkcja stosowana do obróbki obrazu w czasie rzeczywistym. Biblioteka ta wywodzi się od OpenCV. EmguCV jest tzw. wrapperem. Dzięki temu możliwe jest wywoływanie funkcji OpenCV w językach zgodnych z .NET - chociażby C#. Programy wykorzystujące zbiór metod EmguCV może być uruchamiany w systemie

Windows, Linus, Mac OS, iOS i Android. Korzyści oferowane przez tę bibliotekę: odczytywanie obrazu z rejestratora, operacje na kontenerach (zamiast na wektorach z odgórnie ustalonymi rozmiarami), wykrywanie figur geometrycznych, filtracja obrazu, identyfikowanie kolorów pikseli czy też rysowanie na obrazie.

3.3 DroidCam

DroidCam to aplikacja mobilna, która umożliwia przechwytywanie sekwencji obrazów z kamery i podgląd rejestrowanego wideo na komputerze. Komunikacja z komputerem może się odbywać za pośrednictwem sieci lub przy użyciu przewodu USB. Przed rozpoczęciem nagrywania, należy najpierw uruchomić aplikację mobilną. W przypadku komunikacji bezprzewodowej, należy podłączyć telefon i komputer do tej samej sieci. Na ekranie telefonu wyświetli się adres IP z numerem portu, który należy podać na komputerze przy połączeniu z kamerą. Jeżeli użytkownik zdecyduje się na zastosowanie kabla USB, potrzebna będzie aplikacja DroidCam Client na urządzenia desktopowe.

4 Architektura rozwiązania (budowa aplikacji)

System jest podzielony na kilka modułów: moduł analizujący obraz, moduł interpretujący dane wejsciowe z kamery oraz moduł graficzny. System jest oparty o aplikację okienkową napisaną w C#. Jako silnik graficzny wykorzystano Windows Presentation Fundation. Program został podzielony na kilka klas, m.in.: Camera, Detection oraz MovesJumps.

5 Napotkane problemy

Podczas wykonywania zadania projektowego, grupa natrafiła na pewne niedogodności.

5.1 Detekcja planszy do gry

Pierwszym poważniejszym problemem było ustalenie metody detekcji planszy do gry. Początkowo do wykrywania szachownicy używano metody zawartej w bibliotece EmguCV, tj. CvInvoke.FindChessboardCorners. Jednak testowane plansze w większości nie były poprawnie wykrywane przez program. Zamiast tego, zdecydowano się na rozpoznawanie kwadratu i dzielenia go na 8 części w pionie oraz poziomie, dając w rezultacie tablicę o rozmiarach 8x8.

5.2 Identyfikowanie pionków

Drugą nietrywialną sprawę stanowiło identyfikowanie pionków. Dochodziło do tego, iż program nie wykrywał wcale lub też odnajdywał koła w miejscach niewłaściwych, nawet pomimo oświetlenia obserwowanej przestrzeni. Postanowiono odgórnie określić dopuszczalny zakres promienia koła. Następnie, dodano funkcjonalność dzięki której użytkownik może określić wartości minimalne oraz maksymalne wartości kolorów pionków dla obu graczy (ze względu na kompresję obrazu oraz zjawiska takie jak prześwit, cień, niedoświetlenie, itp.) co uskutecznia właściwą detekcję pionków. Odpowiedni dobór wartości kolorów ułatwiają przefiltrowane obrazy kamery - na biało ilustrowane są wybrane obiekty o zadanym kolorze.

5.3 Sprawdzanie poprawności ruchów

Grupa miała problem z pionkami, które mają wiele możliwości ruchów z danego pola. Podjęliśmy próbę rekurencyjnego wywoływania metody sprawdzający czy można wykonać ruchy w prawo

czy w lewo. Podczas testów, wyniki zgadzały się z prawdą.

5.4 Podłączenie modułu sprawdzającego ruchy do aplikacji

Wystąpiły problemy przy próbie dołączenia metod sprawdzających poprawność ruchów do głównej aplikacji.

6 Implementacja

6.1 Wykrywanie pionków

Funkcja wykrywająca okręgi w obrazie. Jest to kluczowa funkcja dla działania całej aplikacji. Wykorzystywana jest do wykrywania pionków na planszy.

```
public static CircleF[]
   GetCircles(Image<Bgr, Byte> img)
{
    UMat uimage =
       ConvertClearImage(img);
    double cannyThreshold = 130.0;
    double
       circleAccumulatorThreshold =
       40.0;
    double dp = 2.0;
    double minDist = 30.0;
    int minRadius = 5;
    int maxRadius = 30;
    CircleF[] circles =
       CvInvoke.HoughCircles(
        uimage, HoughType.Gradient,
           dp, minDist,
           cannyThreshold,
           circleAccumulatorThreshold,
           minRadius, maxRadius);

    return circles;
}
```

6.2 Wykrywanie kolorów

W funkcji wykrywającej kolory pionków wykorzystano następującą funkcję, która pobiera określony piksel obrazu. image.Bitmap.GetPixel((int)circle.Center

(int)circle.Center.Y - i));

6.3 Sprawdzanie poprawności wykonanego ruchu

Część ta składa się z 5 funkcji dla bicia, dla każdego z kolorów pionków, które sprawdzają możliwości bicia każdego znajdującego się na planszy pionka danego koloru, następnie by wspierać też możliwości, a nawet przymus wielobicia, wywołują się rekurencyjnie od nowego położenia pionka.

Następnie jeżeli program nie znalazł żadnej możliwości bicia sprawdza dostępne ruchy nie będące ruchami bijącymi. Ostatecznie wszystkie możliwe konfiguracje planszy po wykonianiu prawidłowego ruchu są zapisywane do listy i porównywane z obecnym stanem planszy decydując czy dany ruch był wykonany poprawnie.

7 Podział prac

Skład grupy stanowi trzech programistów.

Autor i zadanie	
J. Augustyniak	Interfejs graficzny
J. Augustyniak	Wykrywanie planszy
J. Augustyniak	Automatyczne wykrywanie kolorów
J. Augustyniak	Historia ruchów
D. Garczyk	Przechwytywanie obrazu z kamery
D. Garczyk	Wykrywanie pionków na podstawie kolorów
D. Garczyk	Wykrywanie planszy
S. Szeszko	Określenie możliwych ruchów dla pionków gracza oraz bić
S. Szeszko	Wykrywanie planszy

8 Instrukcja użytkowania

8.1 Wymagania

1. Telefon z DroidCam lub kamera
2. Dobre oświetlenie planszy
3. Stabilne podłoże
4. Coś na przykład statywu

8.2 Komunikacja telefon-komputer przez sieć

1. Zainstalować DroidCam na urządzeniu mobilnym Android używanego jako kamera
2. Zainstalować DroidCam Client na komputerze (podczas instalacji zezwolić na instalację kamer na komputerze)
3. Zainstalować sterowniki urządzenia mobilnego na komputerze
4. Zalecane włączenie trybu debugowania USB na urządzeniu mobilnym
5. Upewnić się, że telefon jest podłączony do tej samej sieci co komputer
6. Uruchomić aplikację DroidCam na telefonie
7. Uruchomić DroidCam Client na komputerze
8. Wybrać przycisk z symbolem WiFi
9. Wpisać w Device IP numer adresu IP podany w aplikacji mobilnej DroidCam na urządzeniu rejestrujacym obraz
10. Wpisać w DroidCam Port numer portu podany w aplikacji mobilnej DroidCam na urządzeniu rejestrujacym obraz

11. Upewnij się, iż zaznaczono opcję Video
12. Nacisnąć w desktopowej wersji przycisk Start
13. Po chwili powinien się ukazać obraz rejestrowany z kamery

8.3 Komunikacja telefon-komputer przy użyciu przewodu USB

1. Zainstalować DroidCam na urządzeniu mobilnym Android używanego jako kamera
2. Zainstalować DroidCam Client na komputerze (podczas instalacji zezwolić na instalację kamer na komputerze)
3. Zainstalować sterowniki urządzenia mobilnego na komputerze
4. Zalecane włączenie trybu debugowania USB na urządzeniu mobilnym
5. Podłączyć telefon z komputerem za pośrednictwem kabla USB
6. Upewnić się, że telefon jest odłączony od sieci (WiFi)
7. Uruchomić aplikację DroidCam na telefonie (upewnij się, że podany adrs IP to 0.0.0.0)
8. Uruchomić DroidCam Client na komputerze
9. Wybrać przycisk z symbolem USB
10. Wpisać w DroidCam Port numer portu podany w aplikacji mobilnej DroidCam na urządzeniu rejestrujacym obraz
11. Upewnij się, iż zaznaczono opcję Video
12. Nacisnąć w desktopowej wersji przycisk Start

13. Po chwili powinien się ukazać obraz rejestrowany z kamery

Jeżeli komunikacja przebiega pomyślnie, można uruchomić program.

8.4 Rejestrowanie obrazu z kamery podłączonej przez USB/sieć przy użyciu DroidCam Client

1. Uruchomić plik wykonywalny projektu
2. Upewnić się, że zaznaczono opcję Wybierz kamerę z listy
3. Wybrać z listy rozwijanej kamerę DroidCam Source 3 lub inną zawierającą w nazwie słowo DroidCam
4. Nacisnąć przycisk Rozpocznij

8.5 Wykrywanie szachownicy

1. Kamera powinna znajdować się w miejcu stabilnym (np. na statywie), aby dokładnie zlokalizować szachownicę.
2. Położyć planszę tak, aby kamera mogła zarejestrować cały jej obszar. Plansza:
    - nie może być znacząco odchylona (równe kąty z wszystkich stron),
    - powinna obejmować większość całego przechwytywanego obrazu,
    - położona musi zostać tak, aby lewy górny róg szachownicy był koloru ciemniejszego.
3. Jeżeli na obrazie z kamery wyświetli się pomarańczowy kwadrat w miejscu otoczki planszy oraz zielone kwadraty nakładające się na pola rzeczywistej szachownicy, wówczas detekcja planszy została zrealizowana należycie.

8.6 Wykrywanie pionków

1. Po udanej próbie wykrycia planszy, należy rozstawić pionki obu graczy.

2. Opcjonalnie mozna zaznaczyć opcję "Wykryj kolory pionków".

3. Jeżeli w okienku z rejestrowanym obrazem, wszystkie pionki są otoczone czerwonym kolorem, wtedy detekcja pionków powiodła się. Należy wtedy odznaczyć opcję "Wykryj kolory pionków"

4. W innym przypadku, należy ręcznie przy użyciu suwaków ustawić wartosci kolorów RGB (czerwony, zielony, niebieski) ustawić zakres wykrywanych kolorów pionków dla gracza 1 (zakłada się, że temu graczowi przydzielono ciemniejsze pionki) oraz gracza 1 (jaśniejsze pionki). Właściwie dopasowanie wartości RGB ułatwiają poniższe dwa filtrowane obrazy z kamery. Na biało zilustrowane zostają obiekty o kolorach spełniający wybrany zakres. Należy tak ustawić, aby na biało wyświetlane były tylko figury pionków konkretnego gracza. W innym przypadku, może dojść do zafałszowania danych (np. puste pole może zostać zinterpretowane jako pole, na którym postawiono pionek gracza numer 1).

5. Jeżeli na wyrenderowanej planszy w programie wyswietlają się stabilne pionki na właściwych pozycjach, test został zdany i można rozpocząć rozgrywkę. W przeciwnym razie, należy zmienić wartości suwakami.

8.7 Przebieg gry

1. Gdy wszystkie pionki zostaną poprawnie ustawione na planszy i wykryte przez program można zacząć rozgrywkę

2. Po wykonaniu ruchu należy wcisnąć przycisk i wtedy program nam powie czy ruch był prawidłowy

3. Całość jest zobrazowana na graficznym przedstawieniu planszy, a każdy poprawnie wykonany ruch zapisany do historii ruchów