

Gra w kółko i krzyżyk

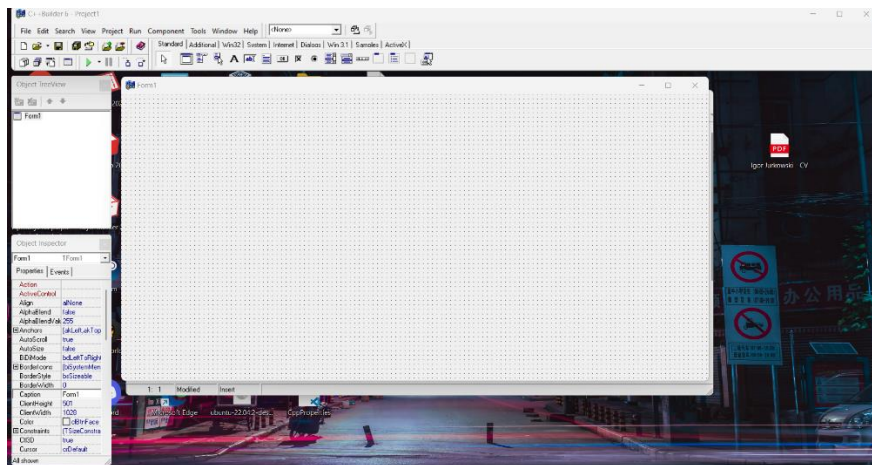
Igor Jurkowski 51865

Podstawy programowania

zorientowane obiektowo

20.05.2023 Rok I semestr II

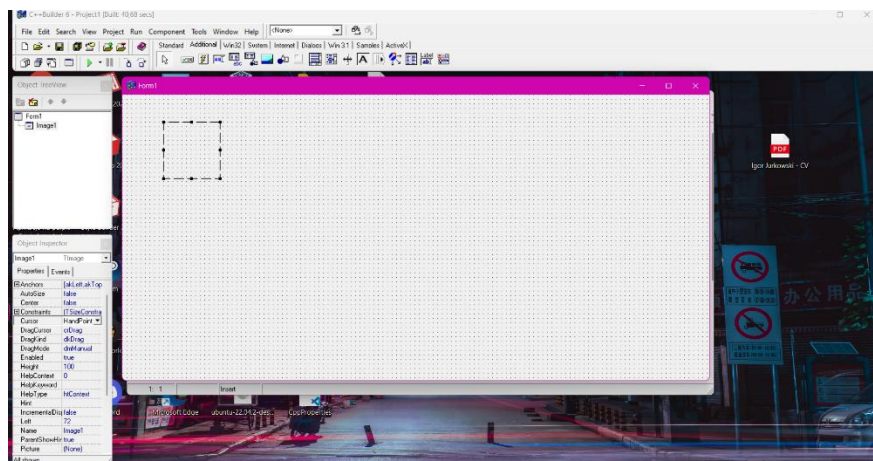
1. Pracę nad grą rozpocząłem od zainstalowania aplikacji C++ Builder, która jest aplikacją RAD co z angielskiego oznacza „szybkie tworzenie aplikacji”.



Obraz 1.1

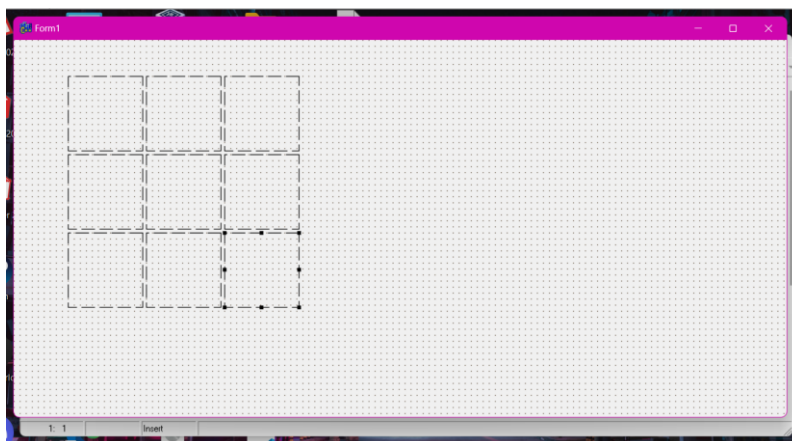
2. Następnym krokiem było pobranie plików graficznych w postaci pliku .bmp potrzebnych do mojej gry.

3. Programowanie interfejsu użytkownika: w zakładce additional dodajemy pole obrazka i wpisujemy rozmiary naszego pliku .bmp o nazwie „pole” oraz ustawiamy kursor na „HandPoint” aby pokazać graczowi że ten element jest aktywny.



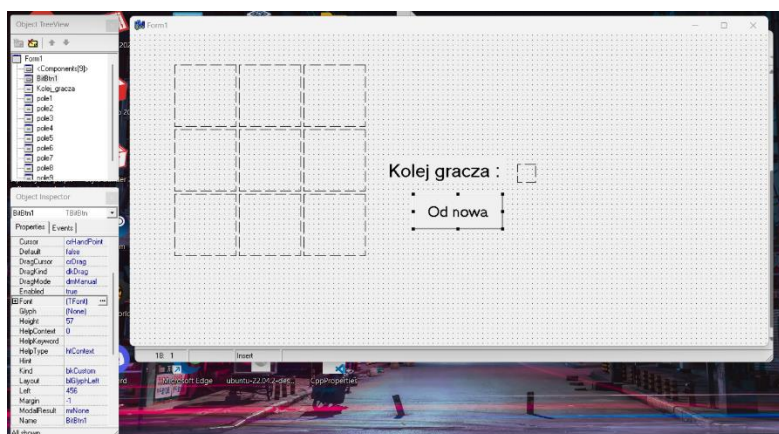
Obraz 1.2

4. Następnym krokiem będzie skopiowanie okienek i ustawienie ich w kształt pola do gry oraz nadajemy każdemu z pól nazwy.



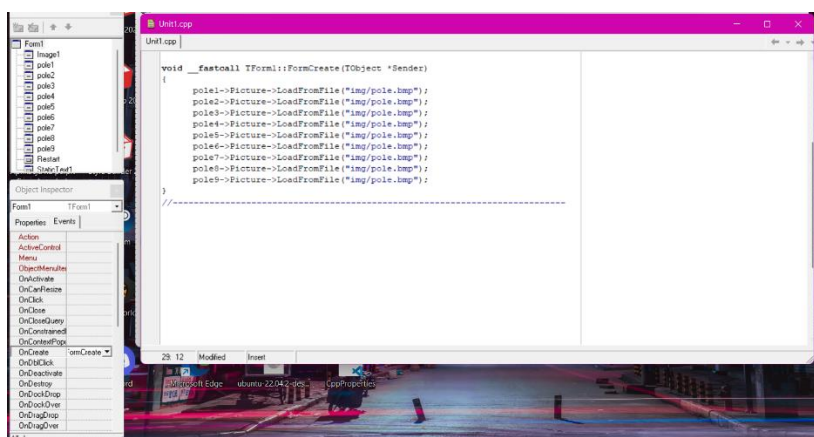
Obraz 1.3

5. Kolejnym krokiem jest dodanie tekstu, wchodzimy w zakładkę standard i naciskamy przycisk wstawiania tekstu. Za tekstem dodajemy pole pokazujące który gracz wykonuje ruch, oraz dodajemy przycisk, który będzie odpowiedzialny za resetowanie planszy.



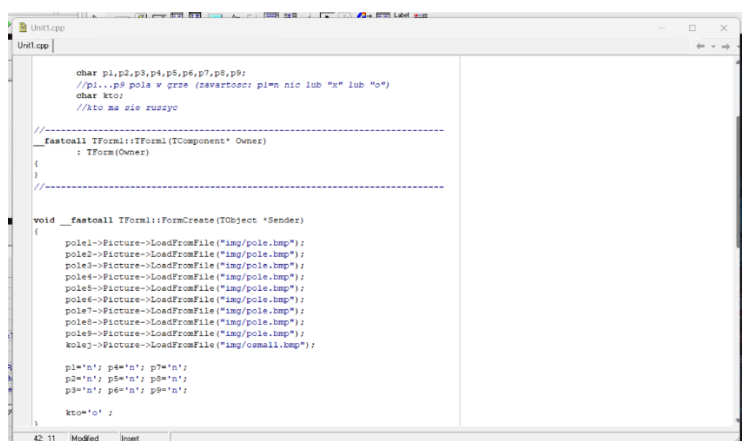
Obraz 1.4

6. Aby przypisać obrazek do „pole1” musimy przejść do zakładki Events i w miejscu OnCreate wpisać FormCreate. Otworzy nam się okno w którym musimy wpisać komendę odpowiedzialną za przypisanie obrazka do „pole1”. Czynność powtarzamy 9 razy dla każdego z pól.



Obraz 1.5

7. Dodajemy 9 zmiennych typu char do przechowywania informacji na temat pól i zmienną „kto” odpowiedzialną za pamiętanie czyja jest tura.



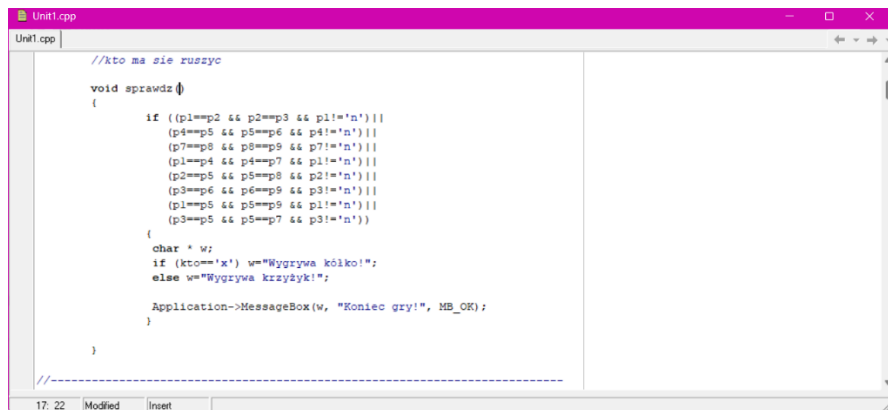
Obraz 1.6

8. Następnym krokiem będzie implementowanie przycisków. Aby ustalić co musi się stać po naciśnięciu na pole musimy dwukrotnie nacisnąć na pole. Pierwsze co należy zrobić to ustalić czy konkretne pole jest puste. Następnie wpisujemy linie dzięki której po kliknięciu na pole1 pojawi się tam kółko. Oraz dwie dodatkowe linie logiczne dla komputera, które pozwolą mu na zapamiętanie wyboru oraz zmianę kółka na krzyżyk w polu pokazującym kolej gracza. Dla estetyki możemy dopisać linię dzięki której znika kursor „łapki” w wybranym już polu. Oraz po komendzie else wklejamy ponownie blok kodu i zmieniamy wartości z kółka na krzyżyk dla drugiego gracza. Czynność powtarzamy dla każdego z pól.



Obraz 1.7

9. Dopusujemy zmienną void odpowiedzialną za sprawdzanie wygranej. Za pomocą funkcji if wypisujemy wszystkie możliwe połączenia pól dające wygraną. Funkcją char z gwiazdką wyświetlamy MessageBoxa pokazującego w okienku informację o wygranej. Aby funkcja sprawdzania działała dodajemy do kodu każdego z przycisków linię „sprawdz();”.



```

//kto ma sie ruszyc

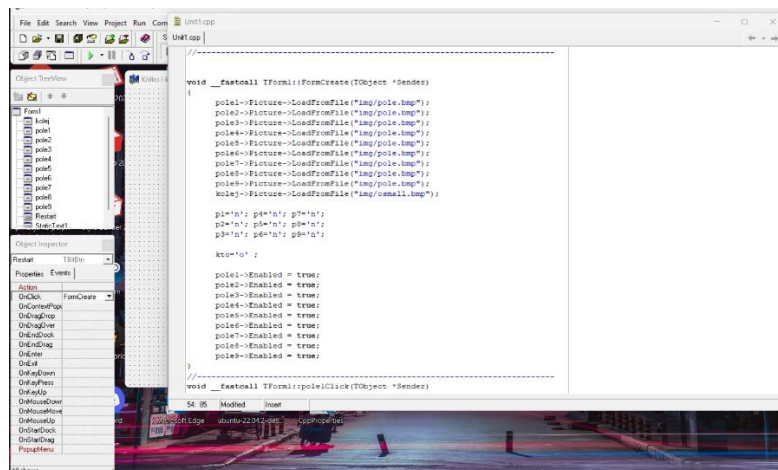
void sprawdz()
{
    if ((p1==p2 && p2==p3 && p1!='n') ||
        (p4==p5 && p5==p6 && p4!='n') ||
        (p7==p8 && p8==p9 && p7!='n') ||
        (p1==p4 && p4==p7 && p1!='n') ||
        (p2==p5 && p5==p8 && p2!='n') ||
        (p3==p6 && p6==p9 && p3!='n') ||
        (p1==p5 && p5==p9 && p1!='n') ||
        (p3==p5 && p5==p7 && p3!='n'))
    {
        char * w;
        if (kto=='x') w="Wygrywa kółko!";
        else w="Wygrywa krzyżyk!";

        Application->MessageBox(w, "Koniec gry!", MB_OK);
    }
}
//-----

```

Obraz 1.8

10. Ostatnim krokiem będzie zaimplementowanie przycisku restartu gry. Aby to zrobić użyjemy pola kodu „FormCreate” i dopisujemy tam linię odpowiedzialną za restart gry, oraz po kliknięciu na przycisk w zakładce Events OnClick ustawiamy z listy FormCreate.



```

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    pole1->Picture->LoadFromFile("img/pole.bmp");
    pole2->Picture->LoadFromFile("img/pole.bmp");
    pole3->Picture->LoadFromFile("img/pole.bmp");
    pole4->Picture->LoadFromFile("img/pole.bmp");
    pole5->Picture->LoadFromFile("img/pole.bmp");
    pole6->Picture->LoadFromFile("img/pole.bmp");
    pole7->Picture->LoadFromFile("img/pole.bmp");
    pole8->Picture->LoadFromFile("img/pole.bmp");
    pole9->Picture->LoadFromFile("img/pole.bmp");
    kolo->Picture->LoadFromFile("img/koala.bmp");

    p1="n"; p2="n"; p3="n";
    p4="n"; p5="n"; p6="n";
    p7="n"; p8="n"; p9="n";

    kto="o";

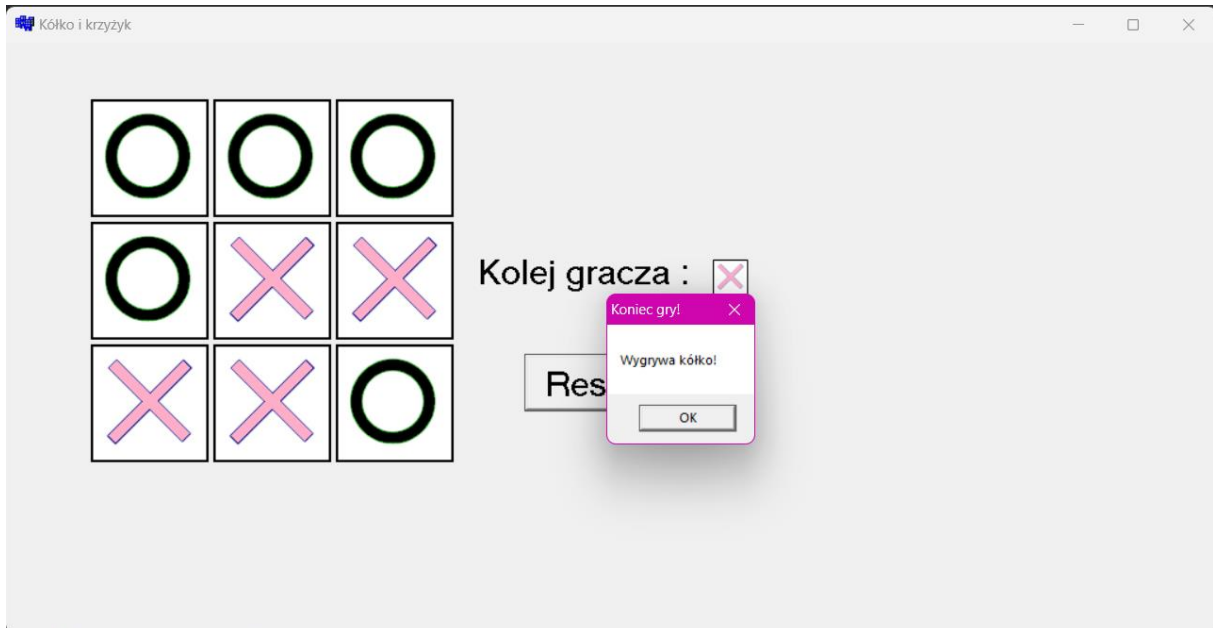
    pole1->Enabled = true;
    pole2->Enabled = true;
    pole3->Enabled = true;
    pole4->Enabled = true;
    pole5->Enabled = true;
    pole6->Enabled = true;
    pole7->Enabled = true;
    pole8->Enabled = true;
    pole9->Enabled = true;
}

void __fastcall TForm1::pole1Click(TObject *Sender)
{
    //-----
}

```

Obraz 1.9

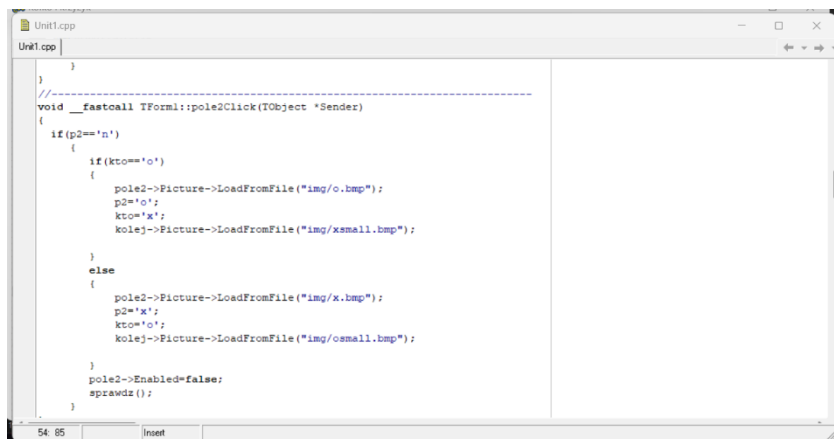
Wygląd gotowej gry



Obraz 2.1

Napotkane problemy

1. Jednym z najczęściej popełnianych błędów w tej aplikacji były źle wpisane numery pól. Stało się to ponieważ używałem funkcji kopiuj wklej, co powodowało że nie zawsze zmieniłem wszystkie numerki w kodzie.



Obraz 3.1

2. Kolejnym błędem który popełniłem było wpisanie linii odpowiedzialnej za anulowanie aktywności pola w złym miejscu. Poprawne miejsce zapisu widać na powyższym screenie.

3. Następnym błędem było niezapisanie funkcji „void sprawdz()” przy każdym z pól. Jej poprawne umiejscowienie również jest widoczne na screenie 3.1

