Raport Końcowy

1.Opis problemu, zadania

Tematem mojego projektu jest gra Tetris

Gra rozpoczyna się na prostokątnej planszy. Plansza ma wymiar 20 wierszy na 16 kolumn w mojej wersji. W trakcie gry pośrodku górnej krawędzi planszy ,pojawiają się pojedynczo klocki ułożone z 4 małych kwadratów nazywanymi też blokami. Klocki te przemieszczają się w kierunku dolnej krawędzi w miarę możliwości. Kiedy jedno tetromino opadnie na samo dno, zostaje unieruchomione, a następne ukazuje się u góry planszy. Gra trwa aż do momentu, w którym klocek nie będzie mógł pojawić się na planszy. Zadaniem gracza jest układanie tetromino na planszy w taki sposób, aby kwadraty składające się na nie utworzyły wiersz na całej szerokości prostokąta. W takiej sytuacji wiersz ten zostaje usunięty, a pozostałe klocki opadają w kierunku dna, tworząc więcej przestrzeni dla następnych elementów.

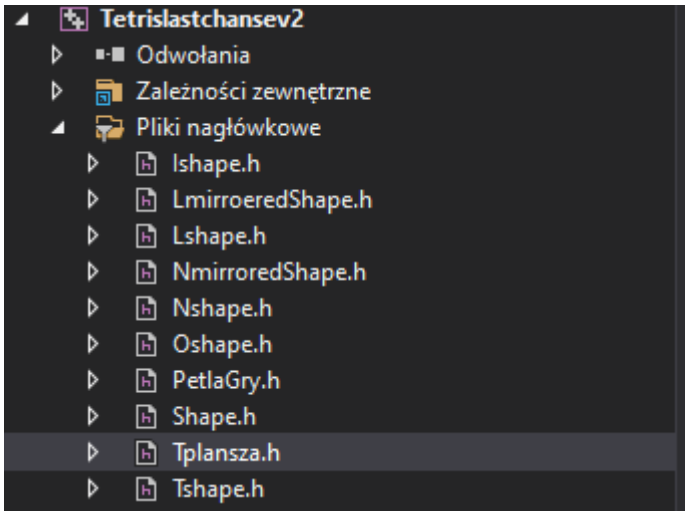
W tetrisie istnieje 7 tetromino:

I,T,O,L,J,S,Z kształtem odpowiadające tym literom.

Bloczkami można poruszać w prawo ,lewo i obracać o 90 stopni zgodnie ze wskazówkami zegara.

2.Podział na obiety/klasy,metody

Podział na klasy wygląda następująco:



Teraz będę opisywał za co odpowiedzialne są klasy

2.1 Tplansza

Zacznijmy od klasy Tplansza.

Klasa Tplansza odpowiedzialna jest za stworzenie planszy i aktualizowanie jej w podczas gry.

Składowe klasy:

```
struct board
{
    bool jest_tetromino;// czy na tym polu jest tetromino
    bool zewnetrzna_warstwa;// czy na tym polu jest zewetrzna warstwa
    bool bloczek_rotacji;
    int wspol_x;// x-owa wspolrzedna
    int wspol_y;// y-owa wspolrzedna
};

class Tplansza
{
public:
    Tplansza();// konstruktor
    void rysuj();// rysuje tablice
    int get_wysokosc(); //ustawia wysokosc planszy
    int get_szerokosc(); //ustawia szerokosc planszy
    int get_pkty();
    int pp_pkty();
    void zewnetrzna_warstwa();
    int zapelniona_linia();
    void linia_w_dol(int wspol_y);
    board plansza[WYSOKOSC_PLANSZY][SZEROKOSC_PLANSZY];//plansza
    int pobierz_X();
    int pobierz_Y();
    int punkty;

private:

    int szerokosc;
    int wysokosc;
    int set_wysokosc(int wys);
    int set_szerokosc(int szer);
    void set_false();
    void set_wspolrzedne();
};
```

Za cała plansze i jej budowę odpowiada ta struktura board teraz tylko wspomnę że ona istnieje przy opisie implementacji opisze to dokładniej.

Tplansza()- W konstruktorze plansza jest tworzona i rysowana

Void rysuj()- Ta metoda jest odpowiedzialna za narysowanie planszy

int get_wysokosc(); służy do pobrania wysokości gdy zajdzie taka potrzeba.

int get_szerokosc(); służy do pobrania szerokości gdy zajdzie taka potrzeba.

int get_pkty(); służy do pobrania punktów gdy zajdzie taka potrzeba.

int pp_pkty(): powiększa punkty o 100.

int zapelniona_linia(); gdy linia jest zapełniona usuwa linie i dodaje punkty.

void linia_w_dol(int wspol_y) Jeżeli linia jest zapełnione to obniża wszystkie klocki powyżej niej o 1.

int pobierz_X(); pobiera współrzędną X (szerokość) aktualnie spadającego tetromino.

int pobierz_Y(); pobiera współrzędną Y(wysokość) aktualnie spadającego tetromino.

Int set_wysokosc(int wys) Ustawia początkową wysokość.

Int set_szerokosc(int wys) Ustawia początkową szerokosc.

Void set_false( ), void set_wspolrzedne() Ustawia początkową współrzędne planszy.

2.2 Shape

Drugą klasą jest klasa Shape odpowiedzialna jest ona za stworzenie na planszy Tetromino i wszelkie operacje za nich takie jak rotacja ,ruch prawo ,lewo ,dół

Wszystkie klasy XShape stworzone są na podstawie klasy Shape napisane są na to samo kopyto zatem opisze tylko jedną klasę Shape reszta wygląda analogicznie.

```
class Shape
{
public:
    virtual void rotacja_klocka(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual int ruch_prawo(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual int ruch_lewo(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual int ruch_dol(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool czy_usmiercic_tetromino(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual void szybki_ruch_dol(int wspol_x, int wspol_y, Tplansza& tab) = 0;
    virtual void usmiercanie_tetromino(int wspol_x, int wspol_y, Tplansza& tab) = 0;
private:
    virtual void rotacja_1(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual void rotacja_2(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual void rotacja_3(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual void rotacja_4(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool pozycja_1(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool pozycja_2(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool pozycja_3(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool pozycja_4(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool mozliwosc_rotacji(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool mozliwosc_ruchu_prawo(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool mozliwosc_ruchu_lewo(int wspol_x, int wspol_y, Tplansza& tab)=0;
    virtual bool mozliwosc_ruchu_dol(int wspol_x, int wspol_y, Tplansza& tab)=0;
```

void rotacja_klocka sprawdza aktualną pozycje klocka ,sprawdza czy można dokonać rotacji i jeżeli można to dokonuje rotacji.

void ruch_prawo sprawdza aktualną pozycje klocka ,sprawdza czy można dokonać ruchu w prawo i jeżeli można to dokonuje ruchu w prawo.

void ruch_lewo sprawdza aktualną pozycje klocka ,sprawdza czy można dokonać ruchu w lewo i jeżeli można to dokonuje ruchu w lewo.

void ruch_dol sprawdza aktualną pozycje klocka ,sprawdza czy można dokonać ruchu w dół i jeżeli można to dokonuje ruchu w dół.

bool czy_usmiercic_tetromino jeżeli nie ma możliwości ruchu dół wtedy zastępuje bloczek rotacji bloczkiem jest_tetromino.

void szybki_ruch_dol robi to samo co ruch_dol tylko robi to do momentu aż ruch w dół już nie jest możliwy

void uśmiercanie_tetromino zastępuje bloczek rotacji bloczkiem jest_tetromino.

2.3 Pętla gry

Klasę pętle gry mogę przyrównać do fizyki w rzeczywistym świecie jest ona odpowiedzialna za działanie gry zawiera Tplansze i Shape* dzięki czemu ta klasa jest już odpowiedzialna za działanie całej gry.

```
class PetlaGry
{
public:
    PetlaGry();
    Shape* natepne_Tetromino;
    Shape* Tetromino;
    Tplansza Plansza;
    int nastepny_klocek;
private:
    void GRA();
    bool koniec_gry();
    void nowy_Shape(int random);
    void nastepny_shape(int random);
    void update_gry();
    void Sterowanie_w();
    void Sterowanie_a();
    void Sterowanie_d();
    void nowy_Tshape();
    void nowy_Oshape();
    void nowy_Ishape();
    void nowy_Nshape();
    void nowy_Lshape();
    void nowy_NmirroredShape();
    void nowy_LmirroredShape();
    int losowanie_liczby();
};
```

PętlaGry tworzy plansze i ogólnie tworzy nową gre.

void GRA() jest to właściwa pętla gry dzięki której gra działa.

bool koniec_gry() Sprawdza czy gra musi się skończyć.

void nowy_shape tworzy nowe tetromino.

void nastepny_shape losuje następny shape względem tego który aktualnie znajduje się na planszy.

void update_gry kasuje obecna plansze i wyświetla nowy stan gry

void sterowanie_w,a,d- przechwytuje odpowiedni klawisz z bufora klawiatury

void nowy_Xshape() tworzy odpowiedni dla siebie shape.

int losowanie)_liczby() losuje liczbę w celu losowego stworzenia nowego tetromino i tetromino +1.

3. Opis podstawowych algorytmów

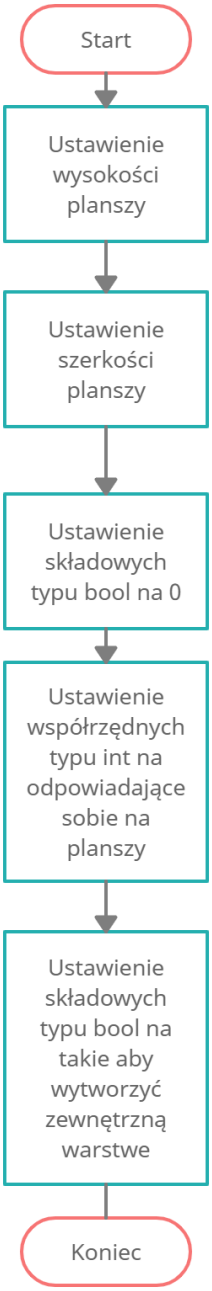
W tym punkcie opisze kilka algorytmów m.in.

-Tworzenie Planszy.

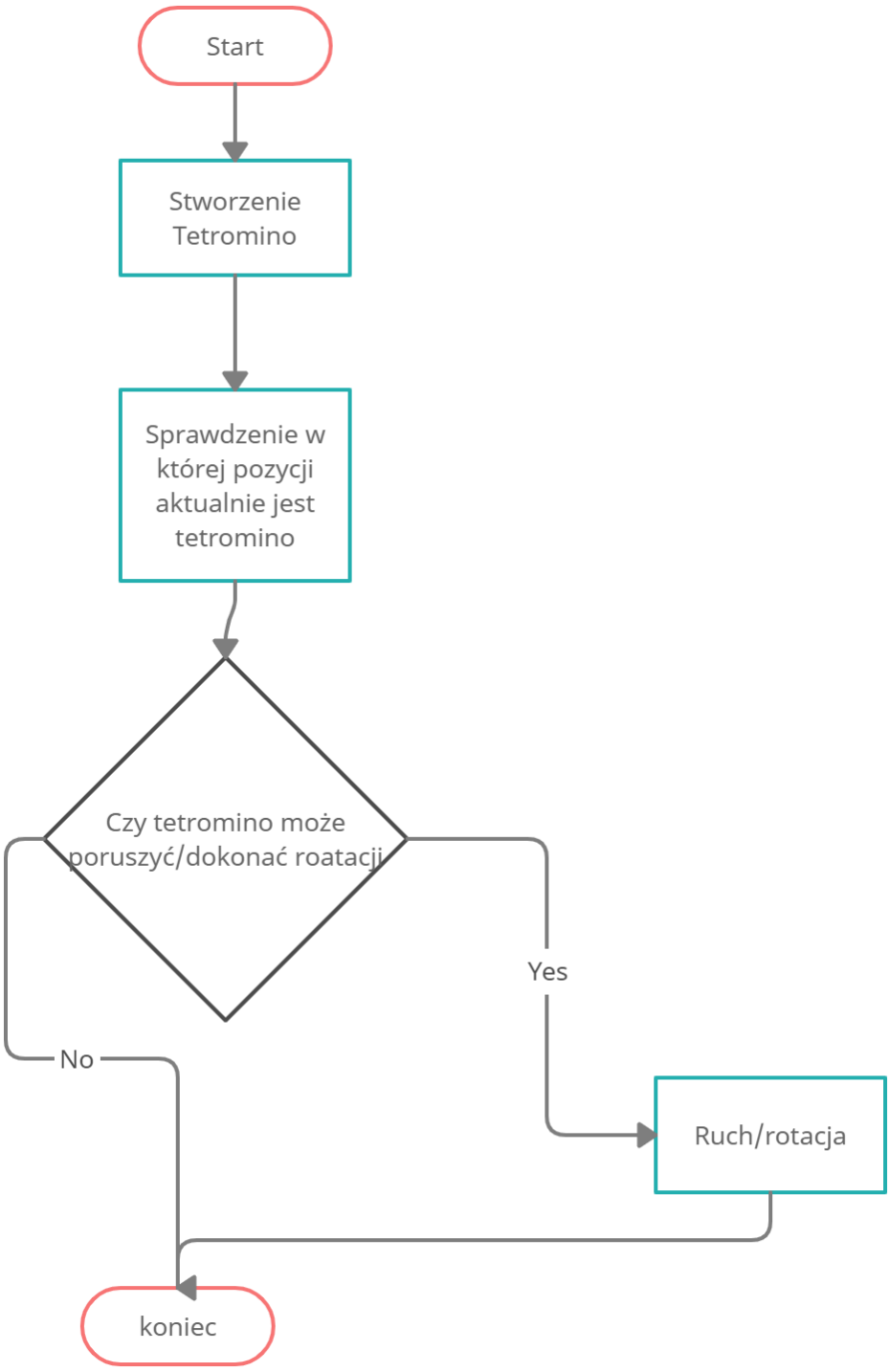
-Ruch Tetromino.

-Pętla gry

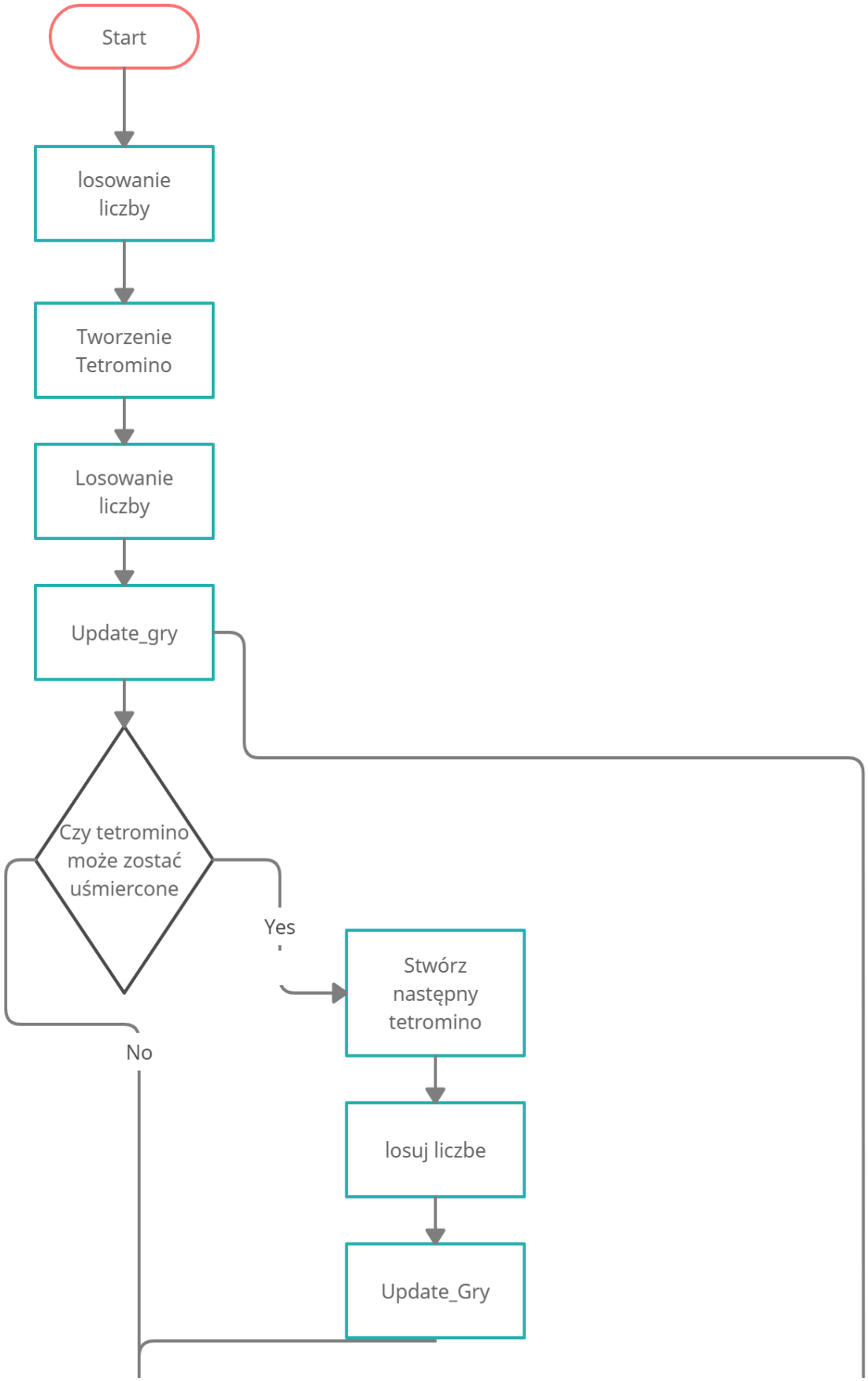
3.1 Tworzenie Planszy

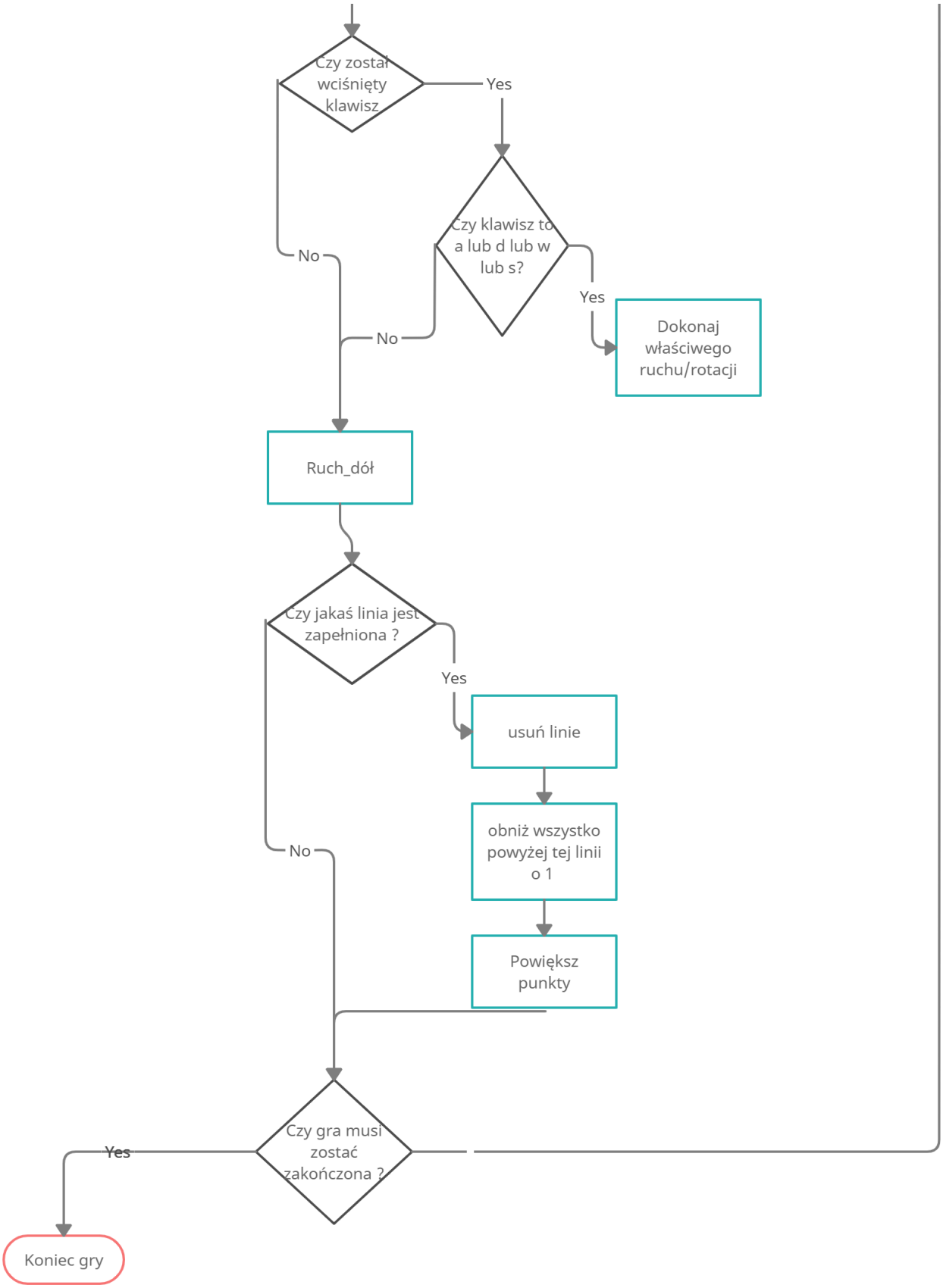


3.2 Ruch Tetromino



3.3 Pętla gry


Start
losowanie liczby
Tworzenie Tetromino
Losowanie liczby
Update_gry
Czy tetromino może zostać uśmiercone
Yes
No
Stwórz następny tetromino
losuj liczbe
Update_Gry

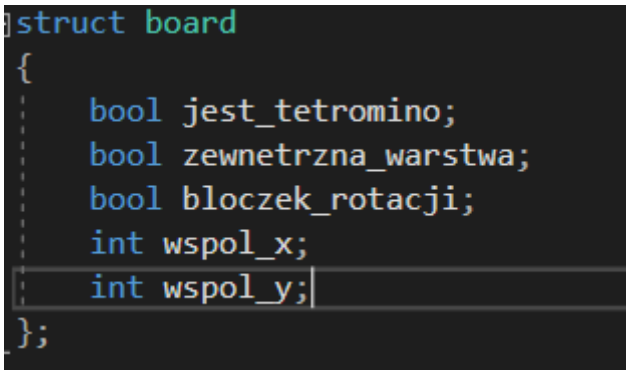

Czy został wciśnięty klawisz
Yes
No
Czy klawisz to a lub d lub w lub s?
Yes
No
Dokonaj właściwego ruchu/rotacji
Ruch_dół
Czy jakaś linia jest zapełniona ?
Yes
No
usuń linie
obniż wszystko powyżej tej linii o 1
Powiększ punkty
Czy gra musi zostać zakończona ?
Yes
Koniec gry

4 Opis implementacji zaprojektowanego systemu

4.1 Tplansza

Ten punkt również zacznę od klasy T-plansza

Tak na prawdę cała plansza opiera się na tej strukturze:



Składa się ona z 5 składowych

bool – jest tetromino zawiera informacje czy na tym polu jest tetomino

bool – zewnętrzna warstwa zawiera informacje czy na tym polu jest krawędź planszy.

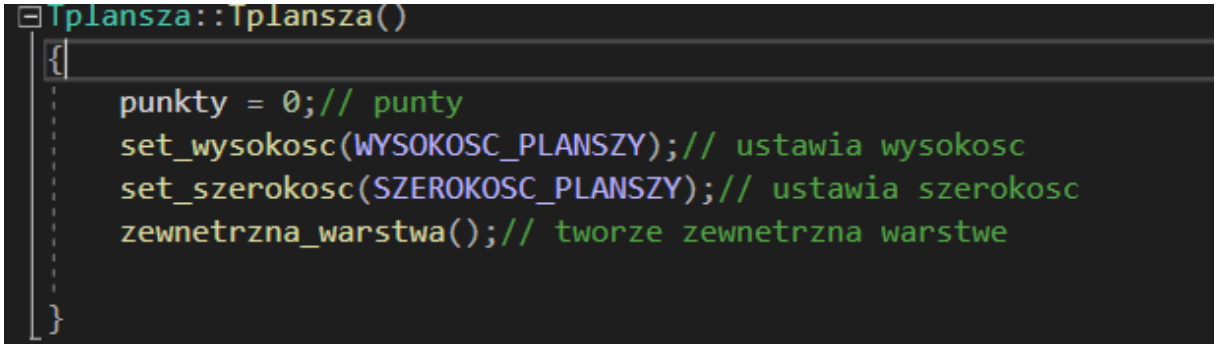
bool – bloczek rotacji zawiera informacje czy na tym polu jest bloczek rotacji

taka mała dygresja bloczek rotacji to też część tetromino ale postanowiłem dodać tą składową dla ułatwienia pisania kodu.

int wspol_x przechowuje w sobie współrzędną X bloczka.

int wspol_y przechowuje w sobie współrzędną Y bloczka.

Konstruktor



Na początku zeruje punkty aby przypadkowo nie została nadana losowa wartość.

Set wysokość i szerokość ustawiają po prostu wysokość i szerokość planszy

Zewnętrzna warstwa opisze w następnym punkcie

Zewnętrzna Warstwa

```
void Tplansza::zewnetrzna_warstwa()// ustawia zewnetrzna warstwe
{
    set_false();
    set_wspolrzedne();
    for (int wysokosc = 0; wysokosc < get_wysokosc(); wysokosc++)
        for (int szerokosc = 0; szerokosc < get_szerokosc(); szerokosc++)
        {
            if (szerokosc == 0)
            {
                plansza[wysokosc][szerokosc].zewnetrzna_warstwa = true;
                plansza[wysokosc][szerokosc].jest_tetromino = false;
            }

            if (szerokosc == get_szerokosc() - 1)
            {
                plansza[wysokosc][szerokosc].zewnetrzna_warstwa = true;
                plansza[wysokosc][szerokosc].jest_tetromino = false;
            }

            if (wysokosc == get_wysokosc() - 1)
            {
                plansza[wysokosc][szerokosc].zewnetrzna_warstwa = true;
                plansza[wysokosc][szerokosc].jest_tetromino = false;
            }
        }
}
```

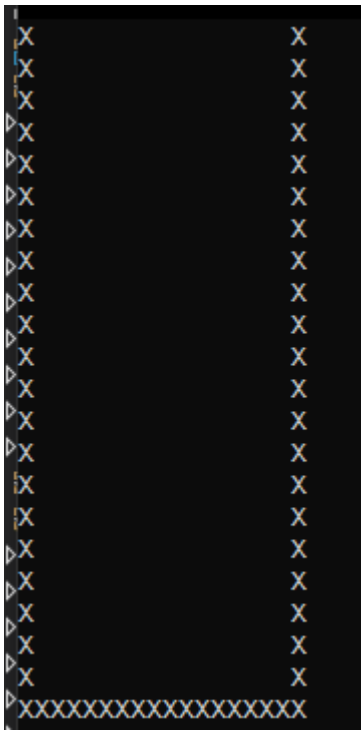
Funkcja set_false – ustawia wszystkie składowe typu bool na false aby nie wkradła się jakaś przypadkowa wartość

```
void Tplansza::set_false()// ustawia jest_tetromino i zewnetrzna_warstwa na falsz na falsz
{
    for (int wysokosc = 0; wysokosc < get_wysokosc(); wysokosc++)
        for (int szerokosc = 0; szerokosc < get_szerokosc(); szerokosc++)
        {
            plansza[wysokosc][szerokosc].zewnetrzna_warstwa = false;
            plansza[wysokosc][szerokosc].jest_tetromino = false;
            plansza[wysokosc][szerokosc].bloczek_rotacji = false;
        }
}
```

Funkcja set_wspolrzedna przypisuje każdemu polu współrzędna X i Y

```
void Tplansza::set_wspolrzedne()// ustawiai wspolrzedne x i y
{
    for (int wysokosc = 0; wysokosc < get_wysokosc(); wysokosc++)
        for (int szerokosc = 0; szerokosc < get_szerokosc(); szerokosc++)
        {
            plansza[wysokosc][szerokosc].wspol_y = wysokosc;
            plansza[wysokosc][szerokosc].wspol_x = szerokosc;
        }
}
```

Ostatnia część zewnętrznej warstwy to zmiana wartości polach znajdujących się na krawędzi odpowiednich wartości aby koniec końców uzyskać:



Rysuj

```
void Tplansza::rysuj()
{
    std::string builder;// string ktory tworzy plansze

    for (int y = 0; y < get_wysokosc(); y++)
    {
        for (int x = 0; x < get_szerokosc(); x++)
        {
            if (plansza[y][x].jest_tetromino)
            {
                builder += "A";
            }
            if (plansza[y][x].bloczek_rotacji)
            {
                builder += "O";
            }

            if (plansza[y][x].zewnetrzna_warstwa)
            {
                builder += "X";
            }

            if (!(plansza[y][x].jest_tetromino || plansza[y][x].zewnetrzna_warstwa|| plansza[y][x].bloczek_rotacji))
            {
                builder += " ";
            }
        }
        builder += "\n";
    }
    std::cout << builder;
}
```

Za samo wyświetlenie planszy odpowiedzialny jest zwykły string który po prostu tworzy plansze.

Zapełniona linia

```
int Tplansza::zapelniona_linia()
{
    int licznik = 0;
    for (int wysokosc = 0; wysokosc < get_wysokosc() - 1; wysokosc++)
    {
        licznik = 0;
        for (int szerokosc = 1; szerokosc < get_szerokosc() - 1; szerokosc++)
        {
            if (plansza[wysokosc][szerokosc].jest_tetromino)
                licznik++;
            if (licznik == (get_szerokosc() - 2))
            {

                for (int szerokosc1 = 1; szerokosc1 < get_szerokosc() - 1; szerokosc1++)
                {
                    plansza[wysokosc][szerokosc1].jest_tetromino = false;

                }
                linia_w_dol(wysokosc);
                return wysokosc;
            }

        }
    }
    return 0;
}
```

Działa to na zasadzie pętli gdy licznik osiągnie pewną wartość kasuje linie i funkcja linia w dól obniża

Wszystko o jedno pole w dół

```
void Tplansza::linia_w_dol(int wspol_y)
{
    for (int i = wspol_y; i >= 5; i--)
        for (int y = 1; y < get_szerokosc() - 1; y++)
        {

            plansza[i + 1][y].jest_tetromino = plansza[i][y].jest_tetromino;
        }


}
```

4.2 Wszystkie Tetromino

Wszystkie tetromino napisane są bardzo podobnie z tą różnicą że mają inny kształt dlatego implementacje pokaże na przykładzie klasy Ishape.

Konstruktor

```
Ishape::Ishape(int wspol_x, int wspol_y, Tplansza& tab)
{
    tab.plansza[wspol_y][wspol_x].jest_tetromino = true;
    tab.plansza[wspol_y + 1][wspol_x].jest_tetromino = true;
    tab.plansza[wspol_y + 2][wspol_x].bloczek_rotacji = true;
    tab.plansza[wspol_y + 3][wspol_x].jest_tetromino = true;
}
```

Na właściwych współrzędnych zmienia odpowiednie składowe.

Pozycja

```
bool Ishape::pozycja_1(int wspol_x, int wspol_y, Tplansza& tab)
{
    if ((tab.plansza[wspol_y][wspol_x].bloczek_rotacji)
        && (tab.plansza[wspol_y - 1][wspol_x].jest_tetromino)
        && (tab.plansza[wspol_y - 2][wspol_x].jest_tetromino)
        && (tab.plansza[wspol_y + 1][wspol_x ].jest_tetromino))
        return true;
    else return false;
}
//*****************************************************************
bool Ishape::pozycja_2(int wspol_x, int wspol_y, Tplansza& tab)
{
    if ((tab.plansza[wspol_y][wspol_x].bloczek_rotacji)
        && (tab.plansza[wspol_y][wspol_x - 1].jest_tetromino)
        && (tab.plansza[wspol_y][wspol_x + 1].jest_tetromino)
        && (tab.plansza[wspol_y][wspol_x + 2].jest_tetromino))
        return true;
    else return false;
}
//*****************************************************************
bool Ishape::pozycja_3(int wspol_x, int wspol_y, Tplansza& tab)
{
    if ((tab.plansza[wspol_y][wspol_x].bloczek_rotacji)
        && (tab.plansza[wspol_y - 1][wspol_x].jest_tetromino)
        && (tab.plansza[wspol_y + 1][wspol_x].jest_tetromino)
        && (tab.plansza[wspol_y + 2][wspol_x].jest_tetromino))
        return true;
    else return false;
}
```

Sprawdza odpowiednie współrzędne względem bloczka rotacji i na podstawie tego sprawdza w jakiej tetromino jest aktualnie pozycji rotacja działa na tej samej zasadzie

Możliwość ruchu prawo,lewo,dół

```
bool Ishape::mozliwosc_ruchu_prawo(int wspol_x, int wspol_y, Tplansza& tab)
{
    if (pozycja_1(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y - 1][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y - 1][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y - 2][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y - 2][wspol_x + 1].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_2(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x + 3].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x + 3].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_3(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y - 1][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y - 1][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 2][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y + 2][wspol_x + 1].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_4(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x + 2].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x + 2].zewnetrzna_warstwa))
            return false;
        else return true;
}
```

```
bool Ishape::mozliwosc_ruchu_lewo(int wspol_x, int wspol_y, Tplansza& tab)
{
    if (pozycja_1(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y - 1][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y - 1][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y - 2][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y - 2][wspol_x - 1].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_2(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x - 2].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x - 2].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_3(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y - 1][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y - 1][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 2][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y + 2][wspol_x - 1].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_4(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y][wspol_x - 3].jest_tetromino)
            || (tab.plansza[wspol_y][wspol_x - 3].zewnetrzna_warstwa))
            return false;
        else return true;
}
```

```
bool Ishape::mozliwosc_ruchu_dol(int wspol_x, int wspol_y, Tplansza& tab)
{
    if (pozycja_1(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y + 2][wspol_x].jest_tetromino)
            || (tab.plansza[wspol_y + 2][wspol_x].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_2(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y + 1 ][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x + 2].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x + 2].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_3(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y + 3][wspol_x].jest_tetromino)
            || (tab.plansza[wspol_y + 3][wspol_x].zewnetrzna_warstwa))
            return false;
        else return true;

    else if (pozycja_4(wspol_x, wspol_y, tab))
        if ((tab.plansza[wspol_y + 1][wspol_x - 2].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x - 2].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x - 1].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x].zewnetrzna_warstwa)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].jest_tetromino)
            || (tab.plansza[wspol_y + 1][wspol_x + 1].zewnetrzna_warstwa))
            return false;
        else return true;
}
```

W tym także nie ma wielkiej filozofii sprawdza w której tetromino jest pozycji a

Później sprawdza czy pola na odpowiednich współrzędnych są zajęte jeśli tak to zwraca fałsz w przeciwnym wypadku prawdę

Ruch

```
int Ishape::ruch_prawo(int wspol_x, int wspol_y, Tplansza& tab)
{
    if (pozycja_1(wspol_x, wspol_y, tab))
    {
        if (mozliwosc_ruchu_prawo(wspol_x, wspol_y, tab))
        {
            tab.plansza[wspol_y][wspol_x].bloczek_rotacji = false;
            tab.plansza[wspol_y - 1][wspol_x].jest_tetromino = false;
            tab.plansza[wspol_y - 2][wspol_x].jest_tetromino = false;
            tab.plansza[wspol_y + 1][wspol_x].jest_tetromino = false;

            tab.plansza[wspol_y][wspol_x + 1].bloczek_rotacji = true;
            tab.plansza[wspol_y - 1][wspol_x + 1].jest_tetromino = true;
            tab.plansza[wspol_y - 2][wspol_x + 1].jest_tetromino = true;
            tab.plansza[wspol_y + 1][wspol_x + 1].jest_tetromino = true;
            return wspol_x + 1;
        }
        else return wspol_x;
    }
    else if (pozycja_2(wspol_x, wspol_y, tab))
    {
        if (mozliwosc_ruchu_prawo(wspol_x, wspol_y, tab))
        {
            tab.plansza[wspol_y][wspol_x].bloczek_rotacji = false;
            tab.plansza[wspol_y][wspol_x + 1].jest_tetromino = false;
            tab.plansza[wspol_y][wspol_x + 2].jest_tetromino = false;
            tab.plansza[wspol_y][wspol_x - 1].jest_tetromino = false;

            tab.plansza[wspol_y][wspol_x + 1].bloczek_rotacji = true;
            tab.plansza[wspol_y][wspol_x + 2].jest_tetromino = true;
            tab.plansza[wspol_y][wspol_x + 3].jest_tetromino = true;
            tab.plansza[wspol_y][wspol_x].jest_tetromino = true;
            return wspol_x + 1;
        }
        else return wspol_x;
```

Sprawdza w której tetromino jest pozycji następnie czy jest możliwość ruchu a jak obydwa te warunki są spełnione kasuje tetromino i zmienia go na odpowiednie współrzędne.

4.3 Pętla gry

Nowy Shape

```
void PetlaGry::nowy_Tshape()
{
    Tetromino = new Tshape(TETROMINOS_START_POSITION_X, TETROMINOS_START_POSITION_Y, Plansza);
    return;
}

void PetlaGry::nowy_Oshape()
{
    Tetromino = new Oshape(TETROMINOS_START_POSITION_X, TETROMINOS_START_POSITION_Y, Plansza);
    return;
}

void PetlaGry::nowy_Ishape()
{
    Tetromino = new Ishape(TETROMINOS_START_POSITION_X, TETROMINOS_START_POSITION_Y, Plansza);
    return;
}

void PetlaGry::nowy_Nshape()
{
    Tetromino = new Nshape(TETROMINOS_START_POSITION_X, TETROMINOS_START_POSITION_Y, Plansza);
    return;
}

void PetlaGry::nowy_Lshape()
{
    Tetromino = new Lshape(TETROMINOS_START_POSITION_X, TETROMINOS_START_POSITION_Y, Plansza);
    return;
}

void PetlaGry::nowy_NmirroredShape()
{
    Tetromino = new NmirroredShape(TETROMINOS_START_POSITION_X, TETROMINOS_START_POSITION_Y, Plansza);
    return;
}
```

Tetromino to wskaźnik typu Shape te funkcje po prostu tworzą dynamicznie nowe tetromino

Update gry

```
void PetlaGry::update_gry()
{
    system("cls");
    Plansza.rysuj();
    std::cout << "\t\t\tWynik :" << Plansza.punkty;
    std::cout << "\n\t\t\tSterowanie: S-szybki ruch dol, W-Rotacja\n\t\t\tA-ruch lewo, A-ruch prawo";
    nastepny_shape(nastepny_klocek);
}
```

Kasuje wszystko na ekranie a następnie rysuje nową plansze wyświetla wynik informacje o sterowaniu i wyświetla następny klocek.

Losowanie liczby

```
int PetlaGry::losowanie_liczby()
{
    srand(time(NULL));
    nastepny_klocek = rand() % 7;
    switch (nastepny_klocek)
    {
    case 0:
        return nastepny_klocek;
        break;
    case 1:
        return nastepny_klocek;
        break;
    case 2:
        return nastepny_klocek;
        break;
    case 3:
        return nastepny_klocek;
        break;
    case 4:
        return nastepny_klocek;
        break;
    case 5:
        return nastepny_klocek;
        break;
    case 6:
        return nastepny_klocek;
        break;
```

Generuje nową liczbę od 0 do 6.

Koniec gry

```
bool PetlaGry::koniec_gry()
{
    for (int i = 1; i < Plansza.get_szerokosc() - 1; i++)
    {
        if (((Plansza.plansza[1][i].bloczek_rotacji)
            ||(Plansza.plansza[0][i].bloczek_rotacji)
            || (Plansza.plansza[2][i].bloczek_rotacji))
            && ((Tetromino->czy_usmiercic_tetromino(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza))))
            return false;

    }
    return true;
```

Sprawdza czy na najwyższych 3 wierszach jest jakiś bloczek i jeżeli on tam się znajduje i funkcja czy uśmiercić tetromino zwróci prawdę wtedy kończy grę

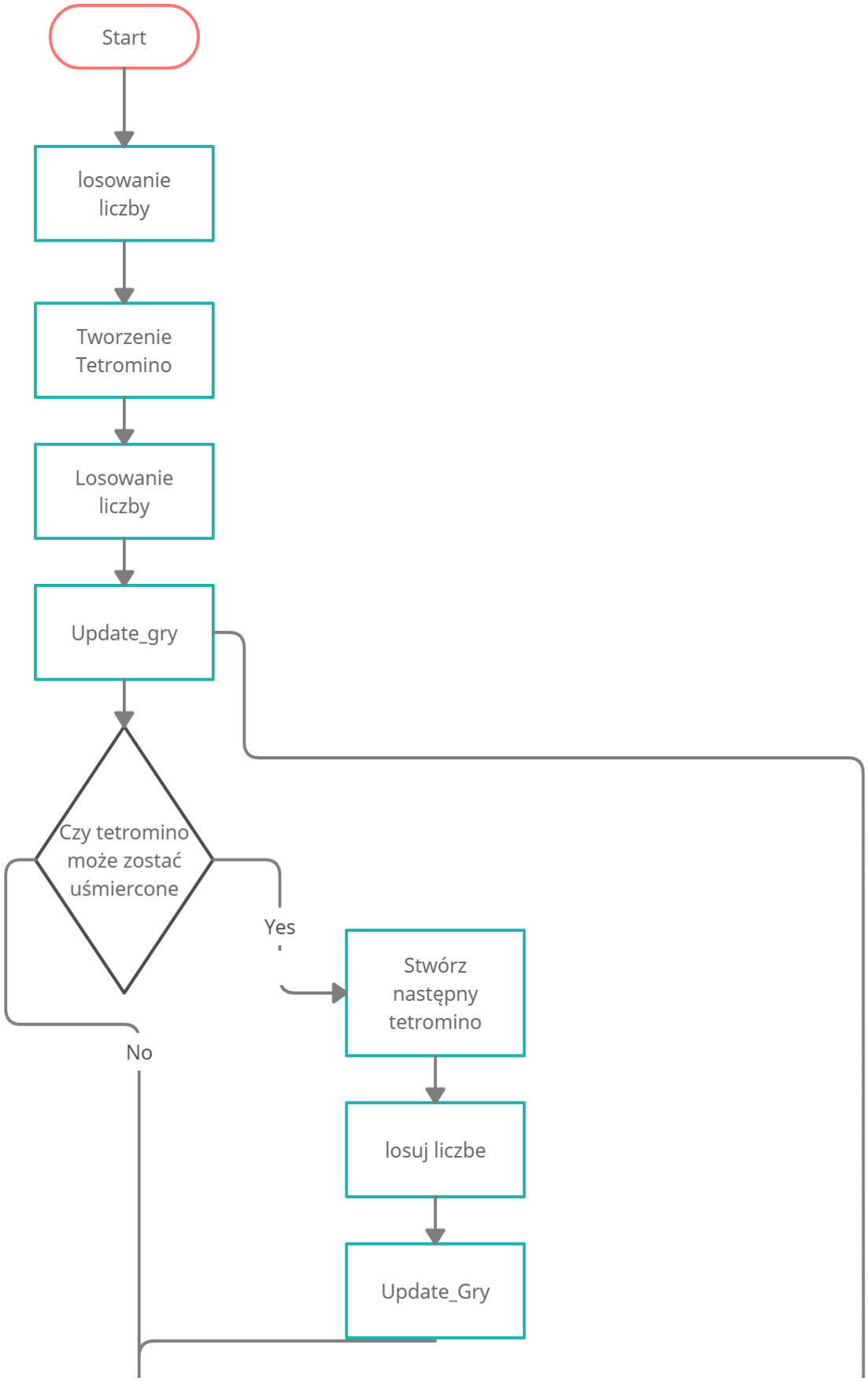
Gra

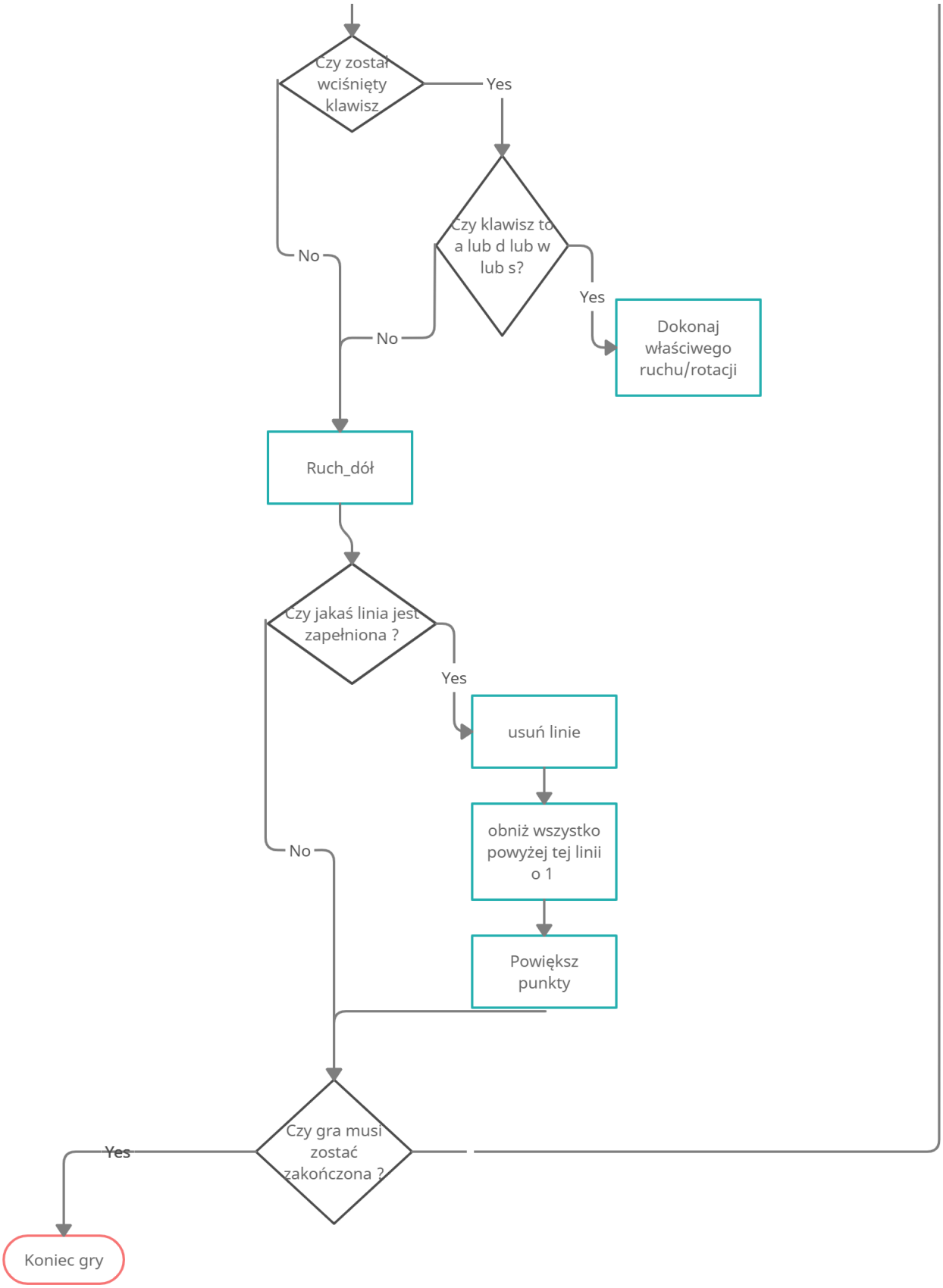
```
char klawisz;
losowanie_liczby();
nowy_Shape(nastepny_klocek);
losowanie_liczby();

while (koniec_gry())
{
    update_gry();
    if (Tetromino->czy_usmiercic_tetromino(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza))
    {
        nowy_Shape(nastepny_klocek);
        losowanie_liczby();
        update_gry();
    }
    if (_kbhit())
    {
        klawisz = _getch();
        if (klawisz == 'a' || klawisz == 'A')
        {
            Tetromino->ruch_lewo(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza);
            update_gry();
        }
        else if (klawisz == 'd' || klawisz == 'D')
        {
            Tetromino->ruch_prawo(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza);
            update_gry();
        }
        else if (klawisz == 's' || klawisz == 'S')
        {
            Tetromino->szybki_ruch_dol(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza);
            update_gry();
        }
        else if (klawisz == 'w' || klawisz == 'W')
        {
            Tetromino->rotacja_klocka(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza);
            update_gry();
        }
        else continue;
    }
    Tetromino->ruch_dol(Plansza.pobierz_X(), Plansza.pobierz_Y(), Plansza);
    if (Plansza.zapelniona_linia())
    {
        Plansza.pp_pkty();
    }
    Sleep(750);
```

Algorytm

Jest zgodny z tym schematem blokowym


Start
losowanie liczby
Tworzenie Tetromino
Losowanie liczby
Update_gry
Czy tetromino może zostać uśmiercone
Yes
No
Stwórz następny tetromino
losuj liczbe
Update_Gry


Czy został wciśnięty klawisz
Yes
No
Czy klawisz to a lub d lub w lub s?
Yes
No
Dokonaj właściwego ruchu/rotacji
Ruch_dół
Czy jakaś linia jest zapełniona ?
Yes
No
usuń linie
obniż wszystko powyżej tej linii o 1
Powiększ punkty
Czy gra musi zostać zakończona ?
Yes
Koniec gry

5 Prosta instrukcja użytkownika

Wystarczyć gre i wszystko dzieje się samo

Sterowanie

A-ruch lewo

D- ruch prawo

W- rotacja klocka

S- szybkie spadniecie klocka

Koniec