Pierwsza gra cd.

Zrealizowaliśmy już część naszej gry i co prawda były to zabiegi istotne, jednak nie zapewniły one nam realizacji żadnej rozgrywki. Tak się stało ponieważ budowaliśmy nasz program z „cegiełek”, której pozwały osiągnąć efekt graficzny, jednak nie pozwoliły na wprowadzenia algorytmu gry. Dzisiejszym zadaniem jest napisanie algorytmu, dzięki któremu możliwe będzie przeprowadzenie rozgrywki pomiędzy zawodnikami.

Pierwszym krokiem jest dodanie metody ustawiającej parametry pocisku:

```
void CPocisk::ustawWartosciPoczatkowe(double predkosc_poczatkowa, double kat){
v0=5*predkosc_poczatkowa; //ustawiamy prędkość  początkową
kat0=kat; // przepisujemy wartość kata pod jakim porusza sie pocisk do obiektu pocisk
vx=v0*cos((kat0-90)*3.14/180);//prędkość w osi x
vy=v0*sin((kat0-90)*3.14/180);//prędkość w osi y
}
```

Obok tej metody przyda się nam metoda resetująca pozycję przycisku:

```
void openGLview::resetujUstawieniePociskow(void){//ustawia pociski w pozycji poczatkowej
pocisk1->px=-120;//pozycja pocisku 1
pocisk1->obudowa->srodek.pozycja[0]=-120;

pocisk1->py=-104;
pocisk1->obudowa->srodek.pozycja[1]=-104;

pocisk2->px=120;//pozycja pocisku 2
pocisk2->obudowa->srodek.pozycja[0]=120;

pocisk2->py=-104;
pocisk2->obudowa->srodek.pozycja[1]=-104;
}
```

Ponieważ będziemy wyświetlali nazwę zwycięscy, dodajmy funkcję, która wyświetla komunikat na ekranie:

```
void openGLview::drawText(int x, int y, char *string) // rysuje napis "string" w pozycji x, y
{
glRasterPos2f(x, y);//pozycja napisu
for (int i = 0; i < strlen (string); i++)//literka po literce
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, string[i]);
};
```

Jej przykładowe wywołanie może wyglądać następująco:

```
char napis[100]="wygral zawodnik 1";
drawText(-20,0,napis);
```

Mając już wszystkie funkcje możemy zaimplementować algorytm gry:

```
        if (etap==1) {//gracz 1 przygotowuje sie do strzalu
//tutaj nic sie nie dzieje, czekamy aż wcisniety zostanie odpowiedni przycisk (jego obsluga w klasie
//CopenGL1Dlg)
        }
        if (etap==2) {//strzela gracz 1 - animacja lotu pocisku
```

```
            pocisk1->trajektoria(wiatr); // zmieniamy pozycje pocisku 1
}
if (pocisk1->obudowa->srodek.pozycja[0]>130)
{//pocisk znajduje sie za prawa krawedzia ekranu
            ///opcja 1: pocisk wraca na scenę z lewej strony
            //pocisk1->px=-130;

            ///opcja 2: kolejka przechodzi na drugiego gracza
            resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
            etap=3; // zmieniamy etap
}
if (pocisk1->obudowa->srodek.pozycja[0]<-130) {//pocisk znajduje sie za lewa krawedzia ekranu
            ///opcja 1: pocisk wraca na scenę z prawej strony
            //pocisk1->px=130;

            ///opcja 2: kolejka przechodzi na drugiego gracza
            resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
            etap=; // zmieniamy etap dopisać
}

if (pocisk1->obudowa->srodek.pozycja[1]<-104) {//pocisk znajduje sie poniżej dolnej linii ekranu
            resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
            etap=3; // zmieniamy etap
}
if (pocisk1->obudowa->srodek.pozycja[1]>90) {//pocisk znajduje sie powyżej  górnej linii ekranu
            resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
            etap=3; // zmieniamy etap
}
if ((pocisk1->obudowa->srodek.pozycja[0]<(armata2->pozycja[0]+19))&&
            (pocisk1->obudowa->srodek.pozycja[0]>(armata2->pozycja[0]+11))&&
            (pocisk1->obudowa->srodek.pozycja[1]<(armata2->pozycja[1]+11))&&
            (pocisk1->obudowa->srodek.pozycja[1]>(armata2->pozycja[1]+3)))
{
            //przeciwnik został trafiony
            etap=; // zmieniamy etap – dopisać
}
if (etap==3) {//gracz 2 przygotowuje sie do strzalu
}
if (etap==4) {//strzela gracz 2 - animacja lotu pocisku
            pocisk2->trajektoria(wiatr); //zmieniamy pozycje pocisku 2
}
if (pocisk2->obudowa->srodek.pozycja[0]>130) {//pocisk znajduje sie za prawa krawedzia ekranu
            //pocisk2->px=-130;

            resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
            etap=1; // zmieniamy etap
}

if (pocisk2->obudowa->srodek.pozycja[0]<-130) {//pocisk znajduje sie za lewa krawedzia ekranu
            //pocisk2->py=130;


            resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
            etap=1; // zmieniamy etap
}
if (pocisk2->obudowa->srodek.pozycja[1]<-104) {//pocisk znajduje sie ponizej dolnej linii ekranu
            resetujUstawieniePociskow(); //resetujemy ustwienia pociskow
```

```
                etap=1; // zmieniamy etap
        }

        if (pocisk2->obudowa->srodek.pozycja[1]>90) {//pocisk znajduje sie powyżej  górnej linii ekranu
                resetujUstawieniePociskow(); //resetujemy ustawienie pociskow
                etap=1; // zmieniamy etap
        }
        if ((pocisk2->obudowa->srodek.pozycja[0]<(armata1->pozycja[0]+19))&&
                (pocisk2->obudowa->srodek.pozycja[0]>(armata1->pozycja[0]+11))&&
                (pocisk2->obudowa->srodek.pozycja[1]<(armata1->pozycja[1]+11))&&
                (pocisk2->obudowa->srodek.pozycja[1]>(armata1->pozycja[1]+3)))
        {   //przeciwnik zostal trafiony
                etap=; //zmieniamy etap – dopisać !
        }
```

Algorytm ten należy wykonywać cyklicznie z odpowiednim okresem (ok. 30ms).

Oczywiście musimy wyświetlić odpowiedni komunikat o wygranej, dodajemy zatem poniższy kod do odpowiedniej funkcji rysującej:

```
if(etap==){
        char napis[100]="Wygral Zawodnik 1";
   drawText(-20,0,napis);
        }

if(etap==){
        char napis[100]="Wygral Zawodnik 2";
   drawText(-20,0,napis);
        }
```

Można się domyślić, że „etap” to pole klasy *openGLview*, które musimy dodać.

Teraz musimy obsłużyć przycisk „Fire”. W odpowiedzi na każde jego klikniecie sprawdzany jest stan algorytmu oraz wybierane są odpowiednie działania:

```
        UpdateData(true);        //konieczne przed odczytywaniem wartości

        if (gl.etap==1) {//sprawdzamy czy ruch nalezy do tego gracza pierwszego
                gl.etap=2; // //jezeli tak to animujemy lot pocisku i zmieniamy etap
                gl.pocisk1->ustawWartosciPoczatkowe(armata1_v0, gl.armata1->kat); //ustawiamy jego
                //wartosci poczatkowe, reszta zostanie wykonana w pliku openGLview.cpp (w funkcji
animacja)
        }
        if (gl.etap==3) {//sprawdzamy czy ruch nalezy do tego gracza drugiego
                gl.etap=4; // //jezeli tak to animujemy lot pocisku i zmieniamy etap
                gl.pocisk2->ustawWartosciPoczatkowe(armata1_v0, gl.armata2->kat); //ustawiamy jego
                //wartosci poczatkowe, reszta zostanie wykonana w pliku openGLview.cpp (w funkcji
animacja)
        }
```

Pamiętając o obsłudze klawiatury musimy dodać odpowiednie warunki, które pozwolą na ustawienie właściwego kąta lufy np.:

```
void openGLview::keyboard(unsigned char key, int x, int y)
{
if (key == 27) { exit(0); }// escape key is ascii 27 }
if (key == 97) {  armata1->kat+=5; } //key a
if (key == 115) { armata1->kat-=5; } //key s

if (key == 102) { armata2->kat+=5; } //key f
if (key == 103) { armata2->kat-=5; } //key g

}
```

Musimy też zmienić dynamikę ruchu naszych pocisków zmieniając parametry w metodzie „trajektoria”:

```
void CPocisk::trajektoria(int wiatr){
double dt=0.01;//krok symulacji lotu pocisku w sekundach
px=px+vx*dt;//obliczamy nowa pozycjê pocisku w osi x - poprzez całkowanie prędkości
py=py+vy*dt;//obliczamy nowa pozycje pocisku w osi y - poprzez całkowanie prędkości

//przepisujemy współrzedną x do kola
//przepisujemy współrzedną y do kola
obudowa->srodek.pozycja[0]=px;
obudowa->srodek.pozycja[1]=py;


vx+=wiatr/100; //zmienimy prędkość w osi x na podstawie siły wiatru
vy-=0.9;//prędkość w osi y maleje w każdej jednostce czasu o stałą wartość
}
```

Teraz można skompilować nasz program. Po uruchomieniu na ekranie powinniśmy uzyskać scenę nie różniącą się od tej z ostatnich zajęć. Pierwszy rozgrywać będzie zawodnik z lewej strony okna (proszę sprawdzić jaki jest ustawiony „etap” w konstruktorze). Jeśli wszystko działa, możemy zmienić położenie początkowe naszych pocisków, tak ,żeby pokrywały się one z korpusami armat (w metodzie: `resetujUstawieniePociskow(void)`).

Teraz możemy już powiedzieć, że nasza gra jest zakończona. Oczywiście pozostawia ona wiele do życzenia, jednak celem naszych zajęć, jest poznanie „podejścia” do problemu. W przyszłości będziecie pisać znacznie bardziej złożone aplikacje w których wykorzystacie zdobyte umiejętności.

Sprawozdanie:

1. Zmodyfikować nasz program, tak aby możliwa była losowa zmiana położenia armat (każda rozgrywka rozpoczyna się od losowania położenia armat – **w osi Y**). Dołączyć kod, który uległ modyfikacji wraz z odpowiednim komentarzem.
2. Chcemy aby podczas rozgrywki wyświetlany komunikat, który informuje o aktualnym rozgrywającym (np. „Teraz rozgrywa: zawodnik 1”). Dołącz kod, który uległ modyfikacji wraz ze stosownym komentarzem.

3. Dodać pole statyczne do odpowiedniej klasy, pozwalające na zliczanie stworzonych obiektów typu "Armata". Dołączyć niezbędny kod. Jak działa pole statyczne?