Sprawozdanie

Programowanie Obiektowe w C++

Gra Ping Pong

- aplikacja konsolowa w języku C++

Wykonał:

Dawid Dłużniewski 12892

Informatyka st., sem. III

Ciechanów 2026

1. Cel i ogólny opis

Gra realizuje klasyczny, dwuosobowy ping-pong w konsoli Windows w języku C++
Logika gry obejmuje pętlę główną, obsługę klawiatury, aktualizację obiektów, wykrywanie kolizji, rysowanie "ekranu" w buforze znakowym oraz zliczanie punktów obu stron.

2. Struktura projektu

- Plik nagłówkowy deklaracje.h:
    - Deklaracja klasy bazowej GameObject.
    - Deklaracja klas pochodnych Paddle (paletka) i Ball (piłka).
    - Deklaracja klasy Game zarządzającej logiką i przebiegiem gry.
    - Deklaracja funkcji pomocniczej hideCursor()
- Plik implementacje.cpp:
    - Implementacje wszystkich metod klas GameObject, Paddle, Ball i Game.
    - Implementacje funkcji pomocniczych: clearScreen() oraz hideCursor()
- Plik main.cpp:
    - Punkt wejścia programu (main()), utworzenie obiektu Game i wywołanie metody run()

3. Opis klas

Klasa GameObject

- Rola: abstrakcyjna klasa bazowa dla wszystkich obiektów gry rysowanych i aktualizowanych w pętli (paletki, piłka)
- Pola:
    - int x, y – aktualna pozycja lewego górnego rogu obiektu w "ekranowej" siatce znaków.
    - int width, height – szerokość i wysokość obiektu w znakach
- Konstruktor:
    - GameObject(int x, int y, int w, int h) – ustawia pozycję i rozmiar obiektu
- Metody wirtualne czysto wirtualne:
    - virtual void update() = 0; – logika aktualizacji obiektu w każdej klatce (ruch, fizyka, reakcje)
    - virtual void draw() = 0; – metoda do rysowania obiektu (tu głównie pomocnicza/testowa, właściwe rysowanie odbywa się w Game::render())
- Gettery:

- int getX() const; int getY() const; int getWidth() const; int getHeight() const; – udostępniają pozycję i rozmiar obiektu na potrzeby kolizji i renderingu

Klasa Paddle

- Dziedziczy po GameObject i reprezentuje pionową paletkę gracza
- Dodatkowe pole:
  - int speed; – prędkość przesuwania paletki w pionie (liczba "linii" na jedną aktualizację wejścia)
- Konstruktor:
  - Paddle(int x, int y, int w, int h, int speed); – wywołuje konstruktor GameObject i ustawia prędkość
- Metody ruchu:
  - void moveUp() – przesunięcie paletki w górę: y -= speed
  - void moveDown() – przesunięcie paletki w dół: y += speed
- Metody:
  - int getCenterY() const; – zwraca współrzędną pionową środka paletki (y + height / 2), przydatną np. do prostego AI.
  - void update() override; – aktualnie pusta, ruch paletki sterowany jest bezpośrednio metodami moveUp/moveDown w reakcji na klawiaturę.
  - void draw() override; – wypisuje diagnostyczny opis paletki w strumieniu cout (nieużywane w finalnym renderze tablicy znaków).

Klasa Ball

- Dziedziczy po GameObject i reprezentuje piłkę poruszającą się po ekranie.
- Dodatkowe pola:
  - int dx, dy; – składowe prędkości piłki w poziomie i pionie (liczba znaków na klatkę).
- Konstruktor:
  - Ball(int x, int y, int size, int dx, int dy); – ustawia pozycję, rozmiar (kwadrat size x size) oraz kierunek ruchu piłki.
- Metody logiki:
  - void update() override; – aktualizuje pozycję: x += dx; y += dy
  - void invertX(); – odwraca poziomą składową prędkości (dx = -dx;), używane przy odbiciu od paletki i przy starcie kolejnej rundy
  - void invertY(); – odwraca pionową składową prędkości (dy = -dy;), używane przy odbiciu od sufitu i podłogi
  - void draw() override; – wypisuje diagnostyczny opis piłki w cout

Klasa Game

- Odpowiada za sterowanie całą grą: pętlę główną, wejście, aktualizację, kolizje i renderowanie
- Stałe pola konfiguracyjne:
    - const int screenWidth = 80;
    - const int screenHeight = 25; – rozmiar wewnętrznego “ekranu gry”, jeden znak to jedna jednostka wymiaru XY
- Stan gry:
    - bool trwaGra; – flaga działania pętli gry.
    - int scoreLeft, scoreRight; – punktacja lewego i prawego gracza
- Obiekty gry:
    - Paddle leftPaddle; – lewa paletka (start przy lewej krawędzi).
    - Paddle rightPaddle; – prawa paletka (start przy prawej krawędzi).
    - Ball ball; – piłka startująca w przybliżonym środku
- Konstruktor:
    - Ustawia wymiary okna, stan trwaGra = true, inicjuje pozycje paletek i piłki oraz zeruje wynik.
- Metody prywatne:
    - void getInput(); – obsługa wejścia z klawiatury:
        - lewa paletka: w (góra), s (dół).
        - prawa paletka: o (góra), l (dół).
        - q – zakończenie gry (ustawia trwaGra = false)
    - void updateGame(); – wywołuje update() na paletkach i piłce (w praktyce realny ruch ma tylko piłka).
    - void checkCollisions(); – obsługa:
        - odbić piłki od sufitu i podłogi (odwrócenie dy i dodatkowe update() dla odsunięcia od krawędzi),
        - ograniczenia ruchu paletek, aby nie wychodziły poza obszar gry (korekta ruchem w przeciwną stronę),
        - kolizji piłki z paletkami (prosta detekcja kolizji prostokąt–punkt i odwrócenie dx),
        - przyznawania punktów, gdy piłka minie lewą lub prawą krawędź okna (inkrementacja scoreLeft lub scoreRight i reset piłki na środek z odpowiednim kierunkiem).

- void render(); – tworzy bufor znakowy i rysuje aktualny stan:
  - Inicjuje dwuwymiarową tablicę znaków wypełnioną spacjami.
  - Rysuje górną i dolną krawędź okna znakiem '-'.
  - Rysuje obie paletki jako pionowe paski '|', na podstawie ich pozycji i wysokości
  - Rysuje piłkę
  - Wypisuje instrukcję sterowania i aktualny wynik.

- Metoda publiczna:
  - void run(); – pętla główna gry:
    - Dopóki trwaGra:
      - wywołuje getInput(),
      - updateGame(),
      - checkCollisions(),
      - render()

Funkcje pomocnicze

- void clearScreen();
  - Używa funkcji Windows API (GetStdHandle, SetConsoleCursorPosition), aby ustawić kursor konsoli na początek, co w połączeniu z ponownym wypisaniem bufora symuluje odświeżanie ekranu bez migania kursora tekstowego
- void hideCursor();
  - Modyfikuje strukturę CONSOLE_CURSOR_INFO, aby ustawić bVisible = FALSE i ukryć kursor w oknie konsoli, poprawiając komfort rozgrywki
- int main();
  - Wywołuje hideCursor(), tworzy obiekt Game game; i uruchamia grę przez game.run();, zwraca 0 po zakończeniu programu