Oblężenie zamku

Poznaliśmy już wszystkie instrukcje zaplanowane na ten kurs języka C. Wykorzystamy je dziś do zilustrowania oblężenia zamku. Twoim zadaniem będzie napisanie programu, który będzie symulował ostrzał zamku przez armaty (kule poruszają się wg. wzorów na rzut ukośny).

Program powinien:

- Otwierać okno graficzne o wymiarach 800×600.
- W pętli głównej:
  - wczytywać z klawiatury wartość prędkości wylotowej kuli V_0, kąta odchylenia lufy od poziomu α oraz położenia armaty na osi x (wylot armaty zawsze ma się znajdować 50 pikseli powyżej poziomu ziemi).
  - ustawiać i rysować schematycznie armatę jako linię (należy sprawdzać czy w danym miejscu da się postawić armatę – mur zamku znajduje się w obszarze $x \geq 550$ (dla uproszczenia radzimy rysować armatę, jako linię o górnym wierzchołku w punkcie $(x, 50)$ i dolnym wierzchołku zależnym od kąta wychylenia).
  - w wewnętrznej pętli służącej do animacji strzału:
    * rysować zamek, który będzie ostrzeliwany (kod funkcji rysującej zamek możesz znaleźć na końcu instrukcji).
    * wykonywać strzał kulą i rysować jej tor za pomocą gęsto rozłożonych okręgów o małym promieniu (użyj funkcji `animate()` w celu uzyskania efektu płynnego ruchu kuli – przykład użycia funkcji `animate()` znajdziesz na końcu instrukcji).
      Tor rzutu ukośnego zadany jest wzorami:

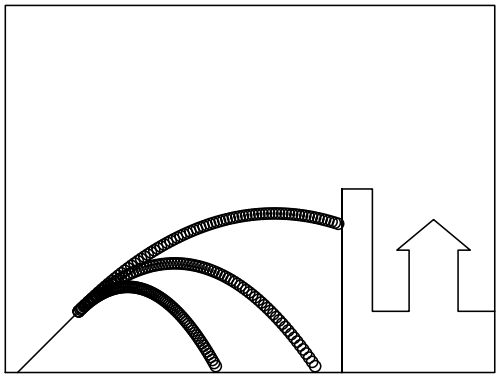
$$x(t) = x_0 + V_0 \cos(\alpha) t$$

$$y(t) = y_0 + V_0 \sin(\alpha) t - \frac{g t^2}{2}.$$

      Program powinien sprawdzać w wewnętrznej pętli czy kula nie uderzyła w pierwszą, pionową ścianę zamku. Jeśli tak, kula ma się na niej zatrzymać, a pętla ulec przerwaniu. W przeciwnym razie, program ma tak długo rysować ruch kuli, aż ta znajdzie się poniżej poziomu ziemi. Ściana zamku zawiera się w przedziale $550 \leq x \leq 600$ a jej górna powierzchnia ma współrzędną $y = 300$.
      Pamiętaj o przeliczeniu współrzędnych fizycznych (pionowej) na współrzędną w układzie współrzędnych ekranu oraz przeliczeniu kąta podawanego w stopniach na kąt podany w radianach na potrzeby funkcji trygonometrycznych.
  - zapytać, po wykonanym strzale, czy udało się trafić w zamek. Jeśli odpowiesz twierdząco, program powinien wyjść z pętli głównej i zakończyć działanie. Jeśli przecząco, program powinien pozwolić wybrać nowe parametry i ponowny strzał.

Oczekiwany wynik

Spodziewamy się, że wynikiem działania Twojego programu będzie rysunek zbliżony do poniższego:



Dodatki

Funkcja rysująca zamek:

```
void RysujZamek() {
  line(550, 300, 550, 600);
  line(550, 300, 600, 300);
  line(600, 300, 600, 500);
  line(600, 500, 660, 500);
  line(660, 500, 660, 400);
  line(660, 400, 640, 400);
  line(640, 400, 700, 350);
  line(700, 350, 760, 400);
  line(760, 400, 740, 400);
  line(740, 400, 740, 500);
  line(740, 500, 800, 500);
}
```

Funkcja `animate()` tak naprawdę jedynie spowalnia wykonywanie pętli `while` tak, aby kółka nie rysowały się zbyt szybko. Argument funkcji to wartość mówiąca o liczbie wykonań ciała pętli w ciągu sekundy. Uwaga: wartość zero mówi o wykonaniu ciała pętli bez opóźnienia.

Przykład jej użycia w naszym programie wygląda następująco:

```
while(1) {                    // petla glowna

  // wczytaj x0, V0 i kat alpha

  while(animate(100)) {       // petla animujaca lot kuli

    // narysuj zamek i armate
    // oblicz nowe wspolrzedne x i y
    // narysuj okrag w odpowiednim miejscu
    // zwieksz parametr t
    // zawrzyj warunek przerwania petli animujacej,
    // gdy kula uderza w sciane lub spada na ziemie

  }

  // zapytaj czy kula trafila w mur i jesli tak
  // wyjdz z petli glownej

}
```