PROJEKT ZALICZENIOWY

Kryspin Kucha IT 23/34

20.12.2023

SYMULACJA FIZYCZNA RZUTU UKOŚNEGO

Możliwość ustawienia podczas działania programu parametrów symulacji:

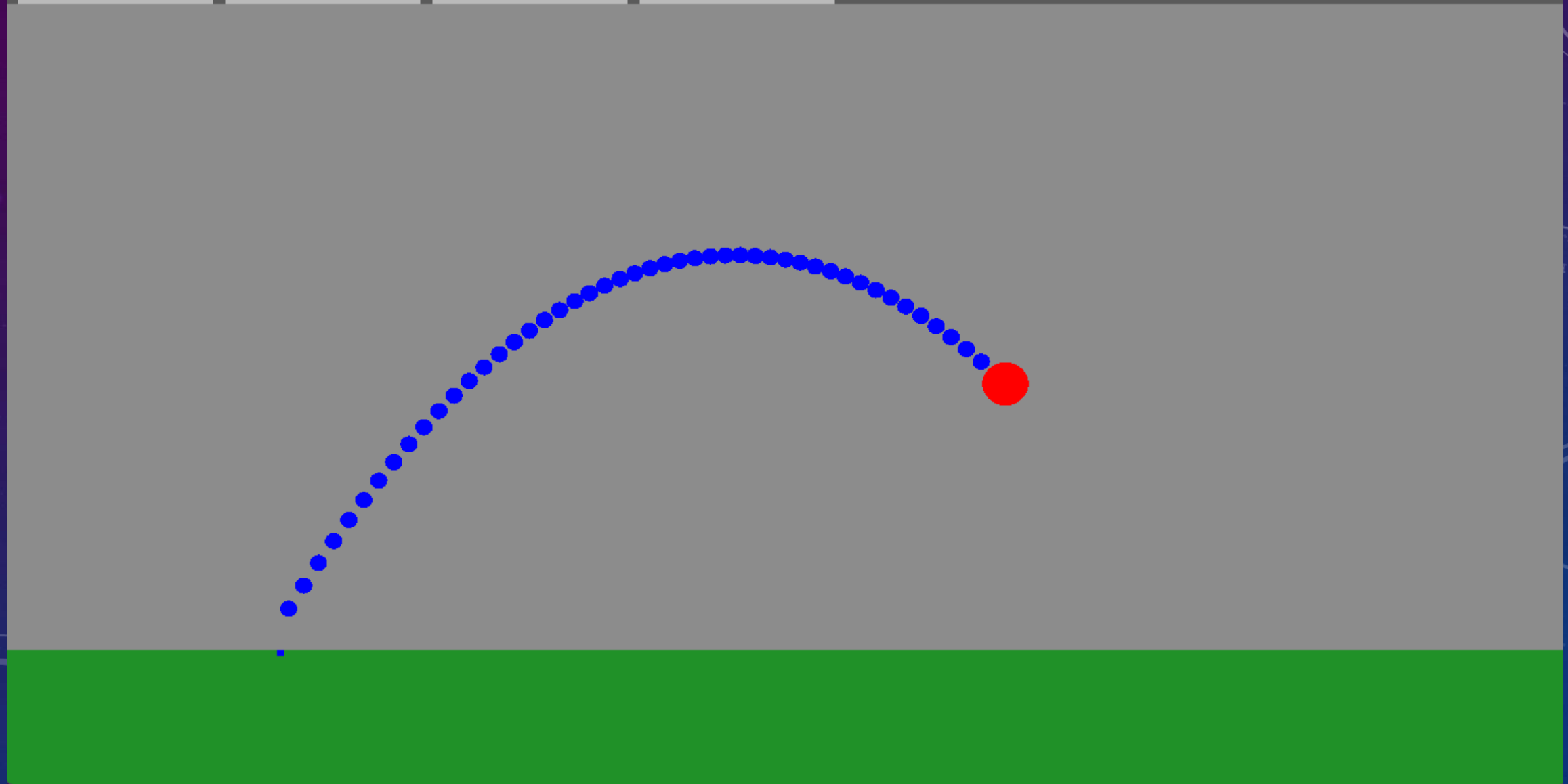
- Szybkości początkowej
- Kątu rzutu
- Wysokości początkowej
- Wartości przyspieszenia grawitacyjnego

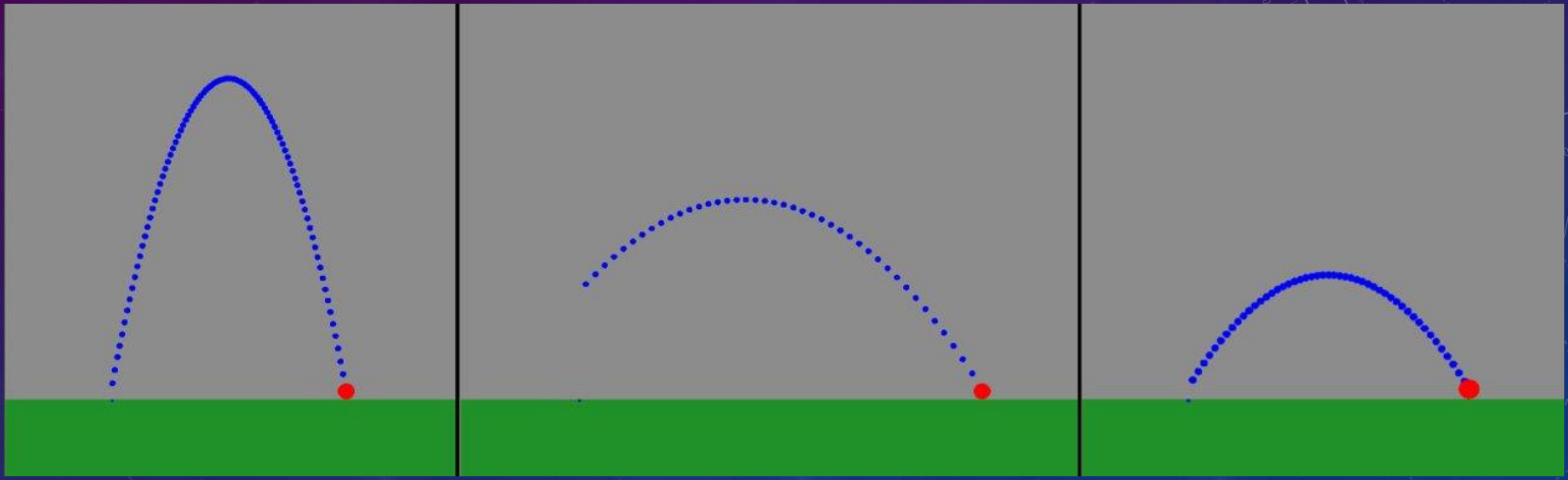
Dodatkowo możliwość ustawienia w pliku config dodatkowych właściwości:

- Koloru tła/piłki/podłoża
- Promienia piłki
- Promienia tracerów
- Nazwy pliku z zapisem statystyk

I innych

V0=12	alfa=60	h=0.00	g=9.81	Zmax=12.71	Hmax=5.50	th=1.06
X=504.39	Y=-180.36	Vx=6.00	Vy=6.10			





Funkcjonalności

- GUI, możliwość ręcznego ustawienia parametrów lotu
- Wyświetlanie obecnej pozycji oraz składowych prędkości V_x V_y w czasie rzeczywistym
- Wyświetlanie końcowych parametrów lotu: zasięg, H_{max} oraz czas osiągnięcia H_{max}
- Rysowanie toru lotu pocisku
- Możliwość zatrzymania/wznowienia/zresetowania symulacji
- Zapis statystyk każdej symulacji do wybranego pliku
- Opcja podążania za piłką oraz statycznego widoku
- Możliwość zmiany widoku oraz przybliżenia/oddalenia go
- Personalizacja wyglądu symulacji
- Inne

Uruchomienie programu

```
14
15  int main()
16  {
17      ProjectileSimulatorArgs config = get_config(CONFIG_PATH);
18      auto simulator = new ProjectileSimulator(config);
19
20      simulator->game_loop();
21
22      return 0;
23  }
24
```

```
482     while (running){
483         deltaTime = deltaClock.restart();
484 
485         while (window.pollEvent(event))
486             handle_event(event);
487 
488         _print_info_to_console();
489 
490         if (simulate_movement)
491         {
492             move();
493             time_for_tracer += deltaTime.asSeconds();
494             if (time_for_tracer >= tracer_interval)
495             {
496                 time_for_tracer = 0.f;
497                 trace();
498             }
499         }
500         update_real_time_widgets();
501 
502         //============ Drawing
503         window.clear(COLOR_BACKGROUND);
504         window.setView(view_game);
505 
506         // Widgets
507         draw_widgets();
508 
509         // Tracers
510         for (auto i = 0; i < tracers.size(); i++)
511             window.draw(*tracers[i]);
512 
513         window.draw(ground);
514         window.draw(start_marker);
515         window.draw(*(ball.getShape()));
516         window.display();
```

Główna pętla symulatora

```
417  void ProjectileSimulator::handle_event(sf::Event event)
418  {
419      switch (event.type){
420      case sf::Event::Closed:
421          running = false;
422          break;
423
424      case sf::Event::KeyPressed:
425
426          handle_moving_view(event);
427
428          if (event.key.code == sf::Keyboard::Tab)
429              handle_tab();
430
431          if (focus_number != -1 and (
432                  event.key.code == sf::Keyboard::BackSpace or
433                  event.key.code == sf::Keyboard::Delete))
434          {
435              widgets_in[focus_number]->delete_last_char();
436          }
437          break;
438
439      case sf::Event::TextEntered:
440
441          if (focus_number == -1)
442              handle_letters(event);
443          else
444              handle_entering_numbers(event);
445          break;
446      }
447  }
```

Obsługa zdarzeń

Ruch pocisku

```
449  void ProjectileSimulator::move()
450  {
451      float dt = deltaTime.asSeconds();
452
453      float xoffset = (vx * dt + ax * dt * dt / 2.f) * unit_to_px;
454      float yoffset = (vy * dt + ay * dt * dt / 2.f) * unit_to_px;
455
456      vx += ax * dt;
457      vy += ay * dt;
458
459      handle_collision(&xoffset, &yoffset);
460
461      ball.move(xoffset, yoffset);
462
463      if (follow_ball)
464          center_view();
465  }
```

Podstawowy binding zmiennych w widgetach

```
112  void Widget::bind_variable(float* var)
113  {
114      binded_variable = var;
115  }
116
117  void Widget::update_variable()
118  {
119      if (binded_variable != NULL and user_text != "")
120      {
121          *binded_variable = std::stof(user_text);
122      }
123  }
124
125  void Widget::update_widget()
126  {
127      if (binded_variable != NULL)
128      {
129          std::stringstream stream;
130          stream << std::fixed << std::setprecision(precision) << *binded_variable;
131
132          user_text = stream.str();
133          this->_set_text();
134      }
135  }
```