Podstawy grafiki komputerwej 1 silnik do gier 2d

Rafał Grot, Kamil Gunia, Piotr Górski

December 1, 2023

1 Wymagania systemowe.

- SFML
- cmake

2 Z wymagań

2.1 Obsługa klawiatury i myszy

Wykorzystaj `setEventHandler()`. Po szczegóły zajrzyj do dokumentacji.

- Przykład

```
#include "engine.hpp"
#include <iostream>

int main() {
  Engine &eng = Engine::getInstance().setWindowTitle("dev").buildWindow();

  eng.setEventHandler(
      Engine::Event::MouseButtonPressed, [](const Engine::Event &ev) {
        std::cout << "Custom event handler Mouse button press "
                  << ev.mouseButton.button << '\t' << ev.mouseButton.x << '\t'
                  << ev.mouseButton.y << "\n";
      });

  eng.loop();
}
```

2.2 Obsługa współrzędnych (Point2D)

Klasa `Point2d`. Po szczegóły zajrzyj do dokumentacji.

2.3 Rysowanie prymitywów

Przy użyciu klass z biblioteki SFML oraz metod `Engine::add()` i `Engine::remove()`.
Szczegóły w dokumentacji technicznej oraz dokumentacji SFML.

2.3.1 Przykład

```
#include "SFML/Graphics/CircleShape.hpp"
#include "engine.hpp"
#include <iostream>

int main() {
  Engine &eng = Engine::getInstance().setWindowTitle("dev").buildWindow();

  sf::CircleShape *circle = new sf::CircleShape(100);

  circle->setPosition(100, 100);
```

```
  eng.add(circle);

  eng.loop();
}
```

2.4 Wypełnianie prymitywów kolorem

Przy użyciu metod obiektów z biblioteki SFML.

2.4.1 Przykład

```
#include "SFML/Graphics/CircleShape.hpp"
#include "SFML/Graphics/Color.hpp"
#include "engine.hpp"
#include <cstdlib>
#include <iostream>
#include <random>

sf::CircleShape *circle = new sf::CircleShape(100);

void customLoop() {
  // count to second
  static float dt{};
  dt += Engine::getInstance().getLastFrameDuration().asSeconds();

  // Do not change color if less then scond elapsed.
  if (dt < 1)
    return;
  dt = 0;

  // Set color based on random RGB values
  sf::Color randomColor{static_cast<sf::Uint8>(rand() % 265),
                        static_cast<sf::Uint8>(rand() % 256),
                        static_cast<sf::Uint8>(rand() % 256)};
  // Set fill color
  circle->setFillColor(randomColor);
}

int main() {
  // initalize random number generator in order to randomize colors.
  srand(time(0));

  Engine &eng =
      Engine::getInstance().setMaxFps(75).setWindowTitle("dev").buildWindow();

  circle->setPosition(100, 100);

  // set custom loop function to change color.
  eng.setLoopFunction(customLoop);

  eng.add(circle);
```

```
  eng.loop();
}
```

2.5 Przekształcenia geometryczne

Przy użyciu SFML.

2.5.1 Przykład

```
#include "SFML/Graphics/RectangleShape.hpp"
#include "engine.hpp"
#include <iostream>

// This is example of transformation usate.
// Consider using AnimatedObject class if you want to do animations based on
// time elapsed

sf::RectangleShape *rect = new sf::RectangleShape({100, 100});

void customLoop() {
  static float dt{};
  dt += Engine::getInstance().getLastFrameDuration().asSeconds();

  // after some time execute step
  if (dt < 0.3)
    return;

  dt = 0;
  static int step{};

  // each second do one of following
  switch (step) {
    // step 0 move right by 100
  case 0:
    rect->move(100, 0);
    ++step;
    break;
  case 1:
    // make rectangle 2 times bigger
    rect->scale(2, 2);
    ++step;
    break;
  case 2:
    // rotate it
    // Check SFML documentation
    rect->setOrigin(rect->getSize().x / 2, rect->getSize().y / 2);

    // rotate by 45 degress
    rect->rotate(45);
    ++step;
```

```
    break;

  case 3:
    // Shrink it
    rect->scale(0.5, 0.5);
    ++step;
    break;
  case 4:
    // Move it to starting position
    rect->move(-100, 0);
    ++step;
    break;

  default:
    // reset step
    step = 0;
  }
}

int main() {
  Engine &eng = Engine::getInstance().setWindowTitle("dev").buildWindow();

  rect->setPosition(100, 300);

  eng.add(rect);

  eng.setLoopFunction(customLoop);

  eng.loop();
}
```

2.6 Obsługa bitmap

Przy użyciu biblioteki SFML tak jak z prymitywami.

2.6.1 Przykład (test8)

```
#include "SFML/Graphics/Sprite.hpp"
#include "engine.hpp"
#include <iostream>

sf::Sprite *sprite = new sf::Sprite{};

int main() {
  Engine &eng = Engine::getInstance().setWindowTitle("dev").buildWindow();

  sf::Texture texture;

  // texture file paths are relative to executable file, not source file.
  if (!texture.loadFromFile("textureFile.png", sf::IntRect(0, 0, 200, 200))) {
    std::cout << "Could not find texture file";
```

```
  }

  sprite->setTexture(texture);
  eng.add(sprite);

  eng.loop();
}
```

2.7 Animowanie bitmap

Przy użyciu klasy `AnimatedSpriteSheet`. Szczegóły w dokumentacji technicznej oraz w dalszej częsci sprawozdania.

```
#include "SFML/Graphics/Color.hpp"
#include "animatedSpriteSheet.hpp"
#include "engine.hpp"
#include <iostream>

namespace G {
std::string basePath = "resources/";
}; // namespace G

AnimatedSpriteSheet animation(G::basePath + "animation");

int main() {
  Engine::getInstance().setMaxFps(3).setResolution({1000, 1000}).buildWindow();

  animation.setPosition({300, 300});
  animation.setColor(sf::Color::Cyan);

  Engine::getInstance().add(&animation);

  Engine::getInstance().loop();
}
```

3 Użycie biblioteki `engine`

Aby mieć dostęp do silnika gry należy wykorzystać plik nagłówkowy `engine.hpp` oraz zlinkować bibliotekę `engine` Klasa Engine jest singletonem. Zaleca się zaposanie z dokumentacją techniczną. Po ustawieniu parametrów należy wywołać metodę `buildWindow()`. Następnie o ile potrzebna wykonać inicjalizację pętli użytkownika. Na koniec należy wywołać metodę silnika `loop()`.

```
int main(){
Engine::getInstance().setMaxFps(3).setResolution({1000, 1000}).buildWindow();

// ...

Engine::getInstance().loop();
}
```

4 Wybrane klasy wykorzystywanie przez silnik.

4.1 Prymitywy

Przy wykorzystaniu Klasa z biblioteki SFML stworzyć obiekty dziedziczące po `sf::Drawable` oraz dodać je do silnika za pomocą metody silnika `Engine::add()`.

4.2 GameObject

GameObject to klasa reprezentująca obiekt w grze.

4.3 Drawable

Drawable jest to abstrakcyjna klasa bazowa wszystkich klas, które można narysować. Dokładnie jest to alias do `sf::Drawable`.

4.4 AnimatedObject

Jest to klasa zawierająca metodę wirtualną `animate()`, należy ją przeciążyć w celu realizacji animacji. Powinna być użyta w dla animacji jeśli inna klasa obsługująca animacje nie jest odpowiednia.

4.5 AnimatedSpriteSheet

Klasa reprezentująca obiekty, które są animowane przy użyciu bitmap. Dziedziczy po AnimatedObject.

- Obiekty tej klasy wczytują informacje o animacji z plików.
- Konstruktor przyjmuje ścieżkę do katalogu, który musi zawierać plik `config.txt`

Metadane animacji opisane w pliku konfiguracyjnym mają następujący format:

```
COMMAND
ARGS
```

Bezpośrednio za linią z `COMMAND` musi znajdować się linia z argumentami oddzielanym spacją. Linie które mają być traktowane jako komentarze zaczynają się znakiem "#".

4.5.1 Command

Prawidłowe kommendy `COMMAND` to:

`SPRITESHEET` Posiada jeden parametr – ścieżkę relatywną od katalogu, w którym znajduje się plik konfiguracyjny, do pliku zawierającego tablice spirtów.

`ANIMATION` Oznacza ładowanie animacji, kolejne wywołania oznaczają nowe typy animacji. Nie przyjmuje parametrów.

`FRAME` Zawiera informacje o pojedynczej klatce animacji. Przyjmuje parametry oznaczające kolejno

- pozycje x lewego górnego rogu sprita.
- pozycje y lewego górnego rogu sprita.
- rozmiar w osi x sprita.
- rozmiar w osi y sprita.
- czas trwania klatki

4.5.2 Przykład

```
#Comment lines start with '#'
#SPRITESHEET should be followed with with relative path in the next line
#(from config file directory) to the spritesheet

#COMMENTS CAN NOT be in between INFO
#example:
#SPRITESHEET
##SOMECOMMENT
#filename.png
#
#is not allowed

#<path>
SPRITESHEET
spritesheet.png

#Indicates new animation
ANIMATION
#FRAME represents frame of animation
#it is followed by line containing
#<pos x> <pos y> <size x> <size y> <duration>
FRAME
0 0 100 100 0.5
FRAME
100 0 100 100 0 0.5
```

4.6 UpdateableObject

Obiekty których stan logiczny się zmienia powinny przeciążać wirtualną metodę `update` klasy `UpdateableObject`. Silnik nie wywołuje tej metody, należy zadbać aby była ona wywoływana np. przy użyciu własnej pętli gry.

5 Przykład użycia silnika

Program wyświetlający animację.

```
#include "SFML/Graphics/Color.hpp"
#include "animatedSpriteSheet.hpp"
#include "engine.hpp"
#include <iostream>

namespace G {
std::string basePath = "resources/";
}; // namespace G

AnimatedSpriteSheet animation(G::basePath + "animation");

int main() {
  Engine::getInstance().setMaxFps(3).setResolution({1000, 1000}).buildWindow();
```

```
  animation.setPosition({300, 300});
  animation.setColor(sf::Color::Cyan);

  Engine::getInstance().add(&animation);

  Engine::getInstance().loop();
}
```