Łukasz Szopa

Nr. Indeksu: 300385

25.04.2020r

Adresat: dr hab. inż. Michał Borecki

# **Zadanie Programistyczne PROE nr. 1**

Specyfikacja wstępna:

1. Temat funkcjonalny programu: Talerz latający względem Słońca
2. Zakres funkcjonalny programu:

Program po pobraniu od użytkownika odpowiednich cech obrazka i należących do niego obiektów tworzy 6 grafik wektorowych w formacie .svg prezentujące lot obiektu, którym jest talerz, względem tła stałego, którym jest Słońce. Talerz podczas lotu zmienia swoje współrzędne położenia oraz nachylenie względem Ziemi.

1. Ogólna dekompozycja zadania na klasy:

|  |  |  |
| --- | --- | --- |
| Klasa | Uwagi | Metody |
| Picture | Zdefiniowane jest pole **unsigned int state** służące do obsługi stanów obrazka | Draw(wirtualna),  Make() – wpisuje do odpowiedniego pliku początkowe linie pliku svg w odpowiedniej rozdzielczości  resolutionChoice(statyczna) – pobiera od użytkownika rozdzielczość tworzonego obrazka i tworzy go |
| Plate:Picture | Kontrola obiektu poprzez zmienną **state** | Draw(wirtualna) – w zależności od wartości zmiennej state, rysunek wykonywany jest wariantowo, talerz na odpowiednich współrzędnych i z odpowiednim nachyleniem  makePlate(statyczna) – pobiera od użytkownika wartości potrzebne do tworzenia talerza a następnie tworzy jego obiekt |
| Sun:Picture |  | Draw (wirtualna) – wykonywany jest rysunek Słońca, na odpowiednich współrzędnych i odpowiednim promieniu  makeSun(statyczna) - pobiera od użytkownika wartości potrzebne do tworzenia słońca a następnie tworzy jego obiekt |
| Animation | Tworzy obiekty innych klas i jest odpowiedzialna za tworzenie grafik | Draw(wirtualna) – wpisuje do pliku svg składnie odpowiadającą poszczególnej klatce  Make() – tworzy obrazki w odpowiednich plikach  DoAll() – tworzy całą animację |

Diagram klas:

Obraz zawierający zrzut ekranu

Opis wygenerowany automatycznie

Opis klas i metod na poziomie funkcjonalnym

1. Picture – Klasa bazowa. Zawiera w sobie zmienną ***unsigned int state***, która opisuje stan lotu talerza. Ma zdefiniowaną metodę ***virtual void draw(std::ofstream& obraz)***, która jest podstawą do dziedziczących po niej klasach, służąca do rysowania odpowiedniego elementu na obrazku. Klasa ma również funkcję statyczną **resolutionChoice()**, która wywoływana na początku programu ustala rozdzielczość, w jakiej wykonane są obrazki.
2. Plate – Klasa pochodna do klasy Picture. Dziedziczy po niej pole ***unsigned int state****,* która opisuje część przebiegu lotu talerza. Przyjmuje ona wartości od 1 do 6. Ponadto, klasa posiada zdefiniowaną metodę ***virtual void draw(std::ofstream& obraz)*** , która służy do rysowania talerza w konkretnym stanie, w pliku, do którego odnosi się zmienna ***std::ofstream& obraz***. Klasa ma funkcję statyczną **makePlate()**, która pobiera od użytkownika wartości potrzebne do utworzenia talerza, oraz zwraca utworzony obiekt.
3. Sun – Klasa pochodna do klasy Picture. Klasa posiada zdefiniowaną metodę ***virtual void draw(std::ofstream& obraz)***, która służy do rysowania Słońca w pliku, którego odnosi się zmienna ***std::ofstream& obraz***. Klasa ma funkcję statyczną **makeSun()**, która pobiera od użytkownika wartości potrzebne do utworzenia Słońca, oraz zwraca utworzony obiekt.
4. Animation – Klasa tworzy obiekty innych klas. Odpowiedzialna jest za tworzenie całego zbioru klatek poprzez funkcję **doAll()**, która wywołuje poszczególne metody innych klas potrzebne do utworzenia obrazka.

Instrukcja posługiwania się programem

Program składa się z 5 plików źródłowych:

1. Main.cpp
2. Picture.h
3. Plate.h
4. Sun.h
5. Animation.h

Kompilacja przez g++:

Komenda: g++ main.cpp && ./a.out

W plikach o rozszerzeniu .h znajdują się definicje odpowiednich klas oraz ich metody. Wszystkie metody oraz nazwy klas są nazwane w języku angielskim. Wyjątkiem są gettery w klasie Picture. Zmienne oraz obiekty nazwane są w języku polskim.

Program po uruchomieniu rozpoczyna pobieranie danych od użytkownika. Jeżeli nie chce on wpisywać żadnych własnych wartości, może posłużyć się wartościami domyślnymi wybierając odpowiednią opcję.

Zostają stworzone obiekty klasy Animation, Picture, Sun i Plate, które wykorzystywane są w późniejszych etapach programu.

Po ustaleniu wszystkich zmiennych wybieranych przez użytkownika, program tworzy plik w formacie svg, do którego później wypisywana jest składnia, która pozwala zaobserwować Słońce oraz talerz, w odpowiednich miejscach.

W pliku main.cpp zawarta jest kontrola generacji kolejnych klatek grafiki. Generacja ta wykonywana jest 6 razy.

**Kod źródłowy programu:**

Main.cpp

#include <iostream>

#include <fstream>

#include <cmath>

#include "Sun.h"

#include "Plate.h"

#include "Picture.h"

#include "Animation.h"

int main()

{

Animation animacja;

animacja.doAll();

}

Picture.h

#ifndef Picture\_h

#define Picture\_h

#include <iostream>

#include <fstream>

int getWidth()

{

int a;

std::cout << "Podaj wybrana szerokosc obrazka: ";

std::cin >> a;

while(a<=0)

{

std::cout << "Podaj dodatnia wartosc" << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

int getHeight()

{

int a;

std::cout << "Podaj wybrana wysokosc obrazka: ";

std::cin >> a;

while(a<=0)

{

std::cout << "Podaj dodatnia wartosc" << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

void closer(std::ofstream& klatka)

{

klatka << "</g>" << std::endl;

klatka << "</svg>" << std::endl;

}

class Picture

{

private:

unsigned int szerokosc;

unsigned int wysokosc;

public:

unsigned int stan;

int getSzer() const {return szerokosc;}

int getWysok() const {return wysokosc;}

virtual void draw(std::ofstream& klatka){}

Picture()

{

stan = 1;

szerokosc = 1080;

wysokosc = 720;

}

Picture(unsigned int szer, unsigned int wysk)

{

szerokosc = szer;

wysokosc = wysk;

std::cout << "Utworzono obrazek o wymiarach: " << szerokosc << "x" << wysokosc << std::endl;

}

void make(std::ofstream& obraz, unsigned int width, unsigned int height)

{

obraz << "<svg width=\"" << width << "\" height=\"" << height << "\"> " << std::endl;

obraz << "<g>" << std::endl;

}

static Picture resolutionChoice();

~Picture(){}

};

Picture Picture::resolutionChoice()

{

Picture rysunek;

int odpowiedz;

std::cout << "Domyslna rozdzielczosc: " << rysunek.getSzer() << "x" << rysunek.getWysok() << std::endl;

std::cout << "Chcesz ja zmienic? 1 - tak \t 0 - nie" << std::endl;

std::cout << "Twoja odpowiedz: ";

std::cin >> odpowiedz;

if(odpowiedz == 0) return rysunek;

if(odpowiedz != 0)

{

int a,b;

a = getWidth();

b = getHeight();

Picture rysunek(a,b);

return rysunek;

}

}

#endif

Plate.h

#ifndef Plate\_h

#define Plate\_h

#include "Picture.h"

int getPlateRadiusX(Picture& rys)

{

int a;

std::cout << "Podaj wybrana dlugosc talerza: ";

std::cin >> a;

while(a<0 || (a>rys.getSzer() && a>rys.getWysok()))

{

std::cout << "Podaj liczbe z zakresu 0 a " << ((rys.getWysok()<rys.getSzer())?rys.getWysok():rys.getSzer()) << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

int getPlateRadiusY(Picture& rys)

{

int a;

std::cout << "Podaj wybrana grubosc talerza: ";

std::cin >> a;

while(a<0 || (a>rys.getSzer() && a>rys.getWysok()))

{

std::cout << "Podaj liczbe z zakresu 0 a " << ((rys.getWysok()<rys.getSzer())?rys.getWysok():rys.getSzer()) << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

int getPlateStartHeight(Picture& rys)

{

int a;

std::cout << "Podaj wybrana wysokosc talerza wzgledem ziemi: ";

std::cin >> a;

while(a<0 || a>rys.getWysok())

{

std::cout << "Podaj liczbe z zakresu 0 a " << rys.getWysok() << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

int getPlateRotation()

{

int a;

std::cout << "Podaj poczatkowe nachylenie talerza (w stopniach): ";

std::cin >> a;

std::cout << "\n";

return a;

}

int getRotationChange()

{

int odpowiedz;

std::cout << "Podaj w skali 1-5 jak szybko ma sie zmieniac nachylenie talerza: ";

std::cin >> odpowiedz;

while(odpowiedz < 1 || odpowiedz > 5)

{

std::cout << "Mozesz tylko podac liczby od 1 do 5. Twoja odpowiedz: ";

std::cin >> odpowiedz;

}

std::cout << "\n";

return odpowiedz;

}

class Plate: public Picture

{

private:

unsigned int szerokosc; // rx

unsigned int wysokosc; // ry

unsigned int wspxt; // cx

unsigned int wspyt; // cy

double nachylenie;

unsigned int zmiana;

public:

Plate()

{

szerokosc = 95;

wysokosc = 20;

wspxt = 100;

wspyt = getWysok()/2;

nachylenie = 20;

zmiana = 5;

}

Plate(unsigned int szer, unsigned int wys, unsigned int x, unsigned int y, double nach, unsigned int zmi)

{

szerokosc = szer;

wysokosc = wys;

wspxt = x;

wspyt = y;

nachylenie = nach;

zmiana = zmi;

}

int getRX() const {return szerokosc;}

int getRY() const {return wysokosc;}

int getCX() const {return wspxt;}

int getCY() const {return wspyt;}

int getRot() const {return nachylenie;}

int getCng() const {return zmiana;}

virtual void draw(std::ostream& obraz)

{

unsigned int tmp = stan - 1;

double kat = 0;

if((nachylenie - (tmp \* ((double)zmiana / 15) \* nachylenie)) > 0) kat = -(nachylenie - (tmp \* ((double)zmiana / 15) \* nachylenie));

else kat = abs(nachylenie - (tmp \* ((double)zmiana / 15) \* nachylenie));

obraz << "<title>Talerz</title>" << std::endl;

obraz << "<ellipse transform=\"rotate(" << kat << " " << wspxt << "," << getSzer() - wspyt << ")\" ";

obraz << "cx=\"" << wspxt + tmp \* getSzer() / 6 << "\" cy=\"" << getWysok() - wspyt << "\" rx=\"" << szerokosc << "\" ry=\"" << wysokosc << "\" style=\"fill:brown\" />" << std::endl;

stan += 1;

}

static Plate makePlate(Picture& rys)

{

Plate temp;

int odpowiedz;

std::cout << "\nWARTOSCI DOMYSLNE TALERZA:" << std::endl;

std::cout << "wysokosc: " << temp.getCY() << ", szerokosc: " << temp.getCX() << ", grubosc: " << temp.getRY() << ", dlugosc: " << temp.getRX() << std::endl;

std::cout << "nachylenie: " << temp.getRot() << ", zmiana nachylenia (skala od 1 do 5): " << temp.getCng() << std::endl;

std::cout << "\nWpisz 1 aby podac cechy talerza lub wpisz 0 aby ustawic wartosci domyslne" << std::endl;

std::cin >> odpowiedz;

if(odpowiedz != 0)

{

Plate temp(getPlateRadiusX(rys), getPlateRadiusY(rys), 100 , getPlateStartHeight(rys), getPlateRotation(), getRotationChange());

return temp;

}

else return temp;

}

~Plate(){}

};

#endif

Sun.h

#ifndef Sun\_h

#define Sun\_h

#include "Picture.h"

int getSunStartHeight(Picture& rys)

{

int a;

std::cout << "Podaj wybrana wysokosc slonca (w pikselach): ";

std::cin >> a;

while(a < ((rys.getWysok())/3) || a > rys.getWysok()-100)

{

if(a < ((rys.getWysok())/3)) std::cout << "Slonce nie moze byc tak nisko, podaj liczbe z zakresu " << rys.getWysok()/3 << " a " << rys.getWysok()-100 << " :" << std::endl;

if((a+100)>(rys.getWysok())) std::cout << "Slonce nie moze wyjsc poza obrazek, podaj liczbe z zakresu " << rys.getWysok()/3 << " a " << rys.getWysok()-100 << " :" << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

int getSunStartWidth(Picture& rys)

{

int a;

std::cout << "Podaj wybrana wspolrzedna x slonca(w pikselach): ";

std::cin >> a;

while(a<100 || a>rys.getSzer()-100)

{

std::cout << "Podaj liczbe z zakresu 100 a " << rys.getSzer()-100 << ": " << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

int getSunRadius(Picture& rys)

{

int a;

std::cout << "Podaj promien slonca (w pikselach): ";

std::cin >> a;

while(a<0 || (a>rys.getSzer() && a>rys.getWysok()))

{

std::cout << "Podaj liczbe z zakresu 0 a " << ((rys.getWysok()<rys.getSzer())?rys.getWysok():rys.getSzer()) << std::endl;

std::cin >> a;

}

std::cout << "\n";

return a;

}

class Sun : public Picture

{

private:

unsigned int promien;

unsigned int wspx;

unsigned int wspy;

public:

int sprawdzPromien(unsigned int a);

virtual void draw(std::ostream& obraz)

{

obraz << "<title>Slonce</title>" << std::endl;

obraz << "<circle id=\"slonce\" cx=\""<< wspx << "\" cy=\""<< getWysok() - wspy << "\" r=\""<< promien << "\"" << " fill=\"#FFFA8D\"/>" << std::endl;

obraz << "<line id=\"l1\" y2=\""<< sprawdzPromien(getWysok() - wspy + ((8\*promien)/3)) << "\" x2=\""<< wspx << "\" y1=\""<< getWysok() - (wspy + (4\*promien/3)) << "\" x1=\""<< wspx << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l2\" y2=\""<< sprawdzPromien(getWysok() - (wspy + (2\*promien))) << "\" x2=\""<< (wspx + 2\*promien) << "\" y1=\""<< getWysok() - (wspy + promien) << "\" x1=\""<< (wspx + promien) << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l3\" y2=\""<< getWysok() - wspy << "\" x2=\""<< (wspx + (8\*promien/3)) << "\" y1=\""<< getWysok() - wspy << "\" x1=\""<< (wspx + (4\*promien)/3) << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l4\" y2=\""<< getWysok() - (wspy - 2\*promien) << "\" x2=\""<< (wspx + 2\*promien) << "\" y1=\""<< getWysok() - (wspy - promien) << "\" x1=\""<< (wspx + promien) << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l5\" y2=\""<< getWysok() - (wspy - ((8\*promien)/3)) << "\" x2=\""<< wspx << "\" y1=\""<< getWysok() - (wspy - (4\*promien/3)) << "\" x1=\""<< wspx << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l6\" y2=\""<< getWysok() - (wspy - (2\*promien)) << "\" x2=\""<< (wspx - 2\*promien) << "\" y1=\""<< getWysok() - (wspy - promien) << "\" x1=\""<< (wspx - promien) << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l7\" y2=\""<< getWysok() - wspy << "\" x2=\""<< (wspx - (8\*promien/3)) << "\" y1=\""<< getWysok() - wspy << "\" x1=\""<< (wspx - (4\*promien)/3) << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

obraz << "<line id=\"l8\" y2=\""<< sprawdzPromien(getWysok() - (wspy + 2\*promien)) << "\" x2=\""<< (wspx - 2\*promien) << "\" y1=\""<< getWysok() - (wspy + promien) << "\" x1=\""<< (wspx - promien) << "\" stroke-width=\"6\" stroke=\"#FFFF91\"/>" << std::endl;

}

int getRadius() const {return promien;}

int getWSPX() const {return wspx;}

int getWSPY() const {return wspy;}

Sun()

{

promien = 120;

wspx = getSzer()/2;

wspy = getWysok() - 200;

}

Sun(unsigned int r, unsigned int x, unsigned int y)

{

promien = r;

wspx = x;

wspy = y;

std::cout << "Utworzono slonce o promieniu r = " << promien << " i wspolrzednych (x,y) : (" << wspx << "," << wspy << ")" << std::endl;

}

static Sun makeSun(Picture& rys)

{

Sun temp;

int odpowiedz;

std::cout << "\nWARTOSCI DOMYSLNE SLONCA:" << std::endl;

std::cout << "wysokosc: " << temp.getWSPY() << ", szerokosc: " << temp.getWSPX() << ", promien: " << temp.getRadius() << std::endl;

std::cout << "\nWpisz 1 aby podac cechy slonca lub wpisz 0 aby ustawic wartosci domyslne" << std::endl;

std::cin >> odpowiedz;

if(odpowiedz != 0)

{

Sun temp(getSunRadius(rys), getSunStartWidth(rys), getSunStartHeight(rys));

return temp;

}

else return temp;

}

~Sun()

{}

};

int Sun::sprawdzPromien(unsigned int a)

{

if(a>getWysok()) return 0;

else return a;

}

#endif

Animation.h

#ifndef Animation\_h

#define Animation\_h

#include "Picture.h"

#include "Plate.h"

#include "Sun.h"

class Animation

{

public:

Animation(){}

Picture rysunek;

Sun slonce;

Plate talerz;

virtual void draw(std::ofstream& klatka, Picture& rysunek, Sun& slonce, Plate& talerz)

{

rysunek.make(klatka, rysunek.getSzer(), rysunek.getWysok());

slonce.draw(klatka);

talerz.draw(klatka);

closer(klatka);

klatka.close();

}

void make(std::ofstream& klatka1, std::ofstream& klatka2, std::ofstream& klatka3, std::ofstream& klatka4, std::ofstream& klatka5, std::ofstream& klatka6, Picture& rysunek, Sun& slonce, Plate& talerz)

{

draw(klatka1, rysunek, slonce, talerz); // klatka1

draw(klatka2,rysunek, slonce, talerz); // klatka2

draw(klatka3,rysunek, slonce, talerz); // klatka3

draw(klatka4,rysunek, slonce, talerz); // klatka4

draw(klatka5,rysunek, slonce, talerz); // klatka5

draw(klatka6, rysunek, slonce, talerz); // klatka6

}

void doAll()

{

rysunek.resolutionChoice();

std::ofstream klatka1 ("klatka1.svg");

std::ofstream klatka2 ("klatka2.svg");

std::ofstream klatka3 ("klatka3.svg");

std::ofstream klatka4 ("klatka4.svg");

std::ofstream klatka5 ("klatka5.svg");

std::ofstream klatka6 ("klatka6.svg");

slonce.makeSun(rysunek);

talerz.makePlate(rysunek);

make(klatka1, klatka2, klatka3, klatka4, klatka5, klatka6, rysunek, slonce, talerz);

}

~Animation(){}

};

#endif

Przykład działania programu:

Obraz zawierający tekst, siedzi, telefon, ulica

Opis wygenerowany automatycznie Pobieranie cech obrazka i obiektów w konsoli

Obraz zawierający zegar, pomiar

Opis wygenerowany automatycznie Klatka nr. 1 Obraz zawierający zegar, pomiar

Opis wygenerowany automatycznie Klatka nr. 2 Obraz zawierający zegar, pomiar

Opis wygenerowany automatycznie Klatka nr. 3 Obraz zawierający zegar

Opis wygenerowany automatycznie Klatka nr. 4 Obraz zawierający zegar, pomiar

Opis wygenerowany automatycznie Klatka nr. 5 Obraz zawierający zegar, pomiar

Opis wygenerowany automatycznie Klatka nr. 6