**Symulator tworzenia sklepu**

Hryhorii Shynkovskyi ID07IO2

Użyto biblioteki LWJGL do tworzenia grafiki <https://www.lwjgl.org/>

**Main Class**

package com.persei;

import org.lwjgl.glfw.\*;

import org.lwjgl.opengl.\*;

import static org.lwjgl.glfw.Callbacks.\*;

import static org.lwjgl.glfw.GLFW.\*;

import static org.lwjgl.opengl.GL11.\*;

import static org.lwjgl.system.MemoryUtil.\*;

public class Main {

private static Shop shop;

private static long window;

private static Model model;

private static String ansver;

public static Thread th;

public static void main(String[] args) {

ShopBuilder shopBuilder = new ShopBuilder();

ShopBuilderDirector shopBuilderDirector = new ShopBuilderDirector();

shop = shopBuilderDirector.constructShop(shopBuilder);

model = new Model(shop);

th = new Thread(runnable);

th.start();

while (true){

System.out.println("Rerun? - y/n ");

ansver = shopBuilderDirector.scanner.next();

if (ansver.compareTo("y") == 0){

model.SetRandomValues();

} else break;

}

// Free the window callbacks and destroy the window

glfwFreeCallbacks(window);

glfwDestroyWindow(window);

// Terminate GLFW and free the error callback

glfwTerminate();

glfwSetErrorCallback(null).free();

System.exit(-1);

}

static Runnable runnable = new Runnable() {

public void run() {

// Setup an error callback. The default implementation

// will print the error message in System.err.

GLFWErrorCallback.createPrint(System.err).set();

// Initialize GLFW. Most GLFW functions will not work before doing this.

if (!glfwInit())

throw new IllegalStateException("Unable to initialize GLFW");

// Configure GLFW

glfwDefaultWindowHints(); // optional, the current window hints are already the default

glfwWindowHint(GLFW\_RESIZABLE, GLFW\_FALSE); // the window will be resizable

// Create the window

window = glfwCreateWindow(800, 800, "Hello World!", NULL, NULL);

if (window == NULL)

throw new RuntimeException("Failed to create the GLFW window");

// Make the OpenGL context current

glfwMakeContextCurrent(window);

// This line is critical for LWJGL's interoperation with GLFW's

// OpenGL context, or any context that is managed externally.

// LWJGL detects the context that is current in the current thread,

// creates the GLCapabilities instance and makes the OpenGL

// bindings available for use.

GL.createCapabilities();

// Set the clear color

glClearColor(0.3f, 0.1f, 0.3f, 0.1f);

// Run the rendering loop until the user has attempted to close

// the window or has pressed the ESCAPE key.

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // clear the framebuffer

glViewport(-200, -200, 800, 800);

model.DrawWalls();

model.DrawEntrance();

model.DrawCounter();

model.DrawShelves();

glfwSwapBuffers(window); // swap the color buffers

// Poll for window events. The key callback above will only be

// invoked during this call.

glfwPollEvents();

}

// Free the window callbacks and destroy the window

glfwFreeCallbacks(window);

glfwDestroyWindow(window);

// Terminate GLFW and free the error callback

glfwTerminate();

glfwSetErrorCallback(null).free();

System.exit(-1);

}

};

}

**Shop Class**

package com.persei;

import java.util.ArrayList;

/\*\*

\* Główna klasa sklepu

\*/

public class Shop {

private int shop\_size\_x;

private int shop\_size\_y;

private WallColor wall\_color;

private int counter\_pos\_x;

private int counter\_pos\_y;

private int entrance\_pos\_x;

private int entrance\_pos\_y;

ArrayList<Shelf> shelfList;

public Shop() {

}

public int getShop\_size\_x() {

return shop\_size\_x;

}

public void setShop\_size\_x(int shop\_size\_x) {

this.shop\_size\_x = shop\_size\_x;

}

public int getShop\_size\_y() {

return shop\_size\_y;

}

public void setShop\_size\_y(int shop\_size\_y) {

this.shop\_size\_y = shop\_size\_y;

}

public WallColor getWall\_color() {

return wall\_color;

}

public void setWall\_color(WallColor wall\_color) {

this.wall\_color = wall\_color;

}

public int getCounter\_pos\_x() {

return counter\_pos\_x;

}

public void setCounter\_pos\_x(int counter\_pos\_x) {

this.counter\_pos\_x = counter\_pos\_x;

}

public int getCounter\_pos\_y() {

return counter\_pos\_y;

}

public void setCounter\_pos\_y(int counter\_pos\_y) {

this.counter\_pos\_y = counter\_pos\_y;

}

public int getEntrance\_pos\_x() {

return entrance\_pos\_x;

}

public void setEntrance\_pos\_x(int entrance\_pos\_x) {

this.entrance\_pos\_x = entrance\_pos\_x;

}

public int getEntrance\_pos\_y() {

return entrance\_pos\_y;

}

public void setEntrance\_pos\_y(int entrance\_pos\_y) {

this.entrance\_pos\_y = entrance\_pos\_y;

}

public ArrayList<Shelf> getShelfList() {

return shelfList;

}

public void setShelfList(ArrayList<Shelf> shelfList) {

this.shelfList = shelfList;

}

}

**Shelf Class**

package com.persei;

/\*\*

\* Główna klasa regałów w sklepie

\*/

public class Shelf {

float shelf\_pos\_x;

float shelf\_pos\_y;

Product product;

public Shelf(int shelf\_pos\_x, int shelf\_pos\_y, Product product) {

this.shelf\_pos\_x = shelf\_pos\_x;

this.shelf\_pos\_y = shelf\_pos\_y;

this.product = product;

}

}

**ShopBuilder Class**

package com.persei;

import java.util.ArrayList;

/\*\*

\* Builder obiektu Shop

\*/

public class ShopBuilder {

private final Shop shop;

public ShopBuilder() {

shop = new Shop();

}

public Shop build() {

return shop;

}

public ShopBuilder setShopSize(int x, int y) {

shop.setShop\_size\_x(x);

shop.setShop\_size\_y(y);

return this;

}

public ShopBuilder setWallColor(WallColor wallColor) {

shop.setWall\_color(wallColor);

return this;

}

public ShopBuilder setCounterPosition(int x, int y) {

shop.setCounter\_pos\_x(x);

shop.setCounter\_pos\_y(y);

return this;

}

public ShopBuilder setEntrancePosition(int x, int y) {

shop.setEntrance\_pos\_x(x);

shop.setEntrance\_pos\_y(y);

return this;

}

public ShopBuilder setShelves(ArrayList<Shelf> shelfList) {

shop.setShelfList(shelfList);

return this;

}

}

**ShopBuilderDirector Class**

package com.persei;

import java.util.ArrayList;

import java.util.Random;

import java.util.Scanner;

/\*\*

\* Klasa kierowania stworzeniem sklepu

\*/

public class ShopBuilderDirector {

public ShopBuilderDirector() {

}

ArrayList<Shelf> shelfList = new ArrayList<Shelf>();

Random rnd = new Random();

int shop\_size\_x, shop\_size\_y;

int counter\_pos;//Pozycja kasy obsługi

Scanner scanner = new Scanner(System.in);

/\*\*

\* Metoda tworzenia sklepu.

\* Rozmiary sklepu są wprowadzane przez użytkownika w konsoli

\* @param shopBuilder

\* @return Shop

\*/

public Shop constructShop(ShopBuilder shopBuilder) {

System.out.println("Input shop X size");

shop\_size\_x = scanner.nextInt();

System.out.println("Input shop Y size");

shop\_size\_y = scanner.nextInt();

shopBuilder.setShopSize(shop\_size\_x, shop\_size\_y);

counter\_pos = rnd.nextInt(shop\_size\_x-1);

shopBuilder.setCounterPosition(counter\_pos, 0);

shopBuilder.setEntrancePosition(counter\_pos+1, 0);

shopBuilder.setWallColor(getRandomWallColor());

fillShelveList();

shopBuilder.setShelves(shelfList);

return shopBuilder.build();

}

/\*\*

\* Zapisywanie losowych produktów do każdego regału w Liscie

\*/

private void fillShelveList() {

for (int y = 0; y < shop\_size\_y; ++y) {

for (int x = 0; x < shop\_size\_x; ++x) {

shelfList.add(new Shelf(x, y, getRandomProduct()));

}

}

}

/\*\*

\* Losowanie produktów z Enum Product

\* @return Product

\*/

private Product getRandomProduct() {

return Product.values()[rnd.nextInt(Product.values().length)];

}

/\*\*

\* Losowanie koloru ścian z Enum WallColor

\* @return WallColor

\*/

private WallColor getRandomWallColor() {

return WallColor.values()[rnd.nextInt(WallColor.values().length)];

}

}

**Model Class**

package com.persei;

import static org.lwjgl.opengl.GL11.\*;

import java.util.ArrayList;

import java.util.Random;

/\*\*

\* Klasa do skalowanie i przechowywania wartości sklepu w postaci do rysowania

\*/

public class Model {

private Random rnd = new Random();

public float scaled\_shop\_size\_x;

public float scaled\_shop\_size\_y;

public float scaled\_entrance\_pos\_x;

public float scaled\_entrance\_pos\_y;

public float scaled\_counter\_pos\_x;

public float scaled\_counter\_pos\_y;

public ArrayList<Shelf> scaled\_shelves;

private float scalenum;

WallColor wallColor;

public Model(Shop shop) {

this.scaled\_shop\_size\_x = (float)shop.getShop\_size\_x();

this.scaled\_shop\_size\_y = (float)shop.getShop\_size\_y();

this.scaled\_entrance\_pos\_x = (float)shop.getEntrance\_pos\_x();

this.scaled\_entrance\_pos\_y = (float)shop.getEntrance\_pos\_y();

this.scaled\_counter\_pos\_x = (float)shop.getCounter\_pos\_x();

this.scaled\_counter\_pos\_y = (float)shop.getCounter\_pos\_y();

this.scaled\_shelves = shop.getShelfList();

this.wallColor = shop.getWall\_color();

getScaled();

}

/\*\*

\* skalowanie wszystkich wartości

\*/

private void getScaled(){

if (scaled\_shop\_size\_x>scaled\_shop\_size\_y){

scalenum = scaled\_shop\_size\_x;

} else {

scalenum = scaled\_shop\_size\_y;

}

scaled\_shop\_size\_x /=scalenum;

scaled\_shop\_size\_y /=scalenum;

scaled\_entrance\_pos\_x /=scalenum;

scaled\_entrance\_pos\_y /=scalenum;

scaled\_counter\_pos\_x /=scalenum;

scaled\_counter\_pos\_y /=scalenum;

for (int i = 0 ; i<scaled\_shelves.size(); ++i){

scaled\_shelves.get(i).shelf\_pos\_x /= scalenum;

scaled\_shelves.get(i).shelf\_pos\_y /= scalenum;

}

}

/\*\*

\* metoda do definiowania koloru z Enum WallColor

\* @return float[r,g,b]

\*/

private float[] DefineWallColor(){

if (wallColor == WallColor.White){

return new float[]{ 1f,1f,1f};

}

if (wallColor == WallColor.Black){

return new float[]{ 0f,0f,0f};

}

if (wallColor == WallColor.Red){

return new float[]{ 1f,0f,0f};

}

if (wallColor == WallColor.Green){

return new float[]{ 0f,1f,0f};

}

else return new float[]{ 0f,0f,1f};

}

/\*\*

\* Rysowanie ścian w oknie OpenGL

\*/

public void DrawWalls(){

glBegin(GL\_LINE\_STRIP);

glColor3fv(DefineWallColor());

glVertex2f(0f,0f);

glVertex2f(scaled\_shop\_size\_x,0f);

glVertex2f(scaled\_shop\_size\_x,scaled\_shop\_size\_y);

glVertex2f(0f,scaled\_shop\_size\_y);

glVertex2f(0f,0f);

glEnd();

}

/\*\*

\* Rysowanie wejścia w oknie OpenGL

\*/

public void DrawEntrance(){

glBegin(GL\_QUADS);

glColor3f(0.5f,0.5f,0.2f);

glVertex2f(scaled\_entrance\_pos\_x,scaled\_entrance\_pos\_y);

glVertex2f(scaled\_entrance\_pos\_x+0.5f/scalenum,scaled\_entrance\_pos\_y);

glVertex2f(scaled\_entrance\_pos\_x+0.5f/scalenum,scaled\_entrance\_pos\_y-0.05f/scalenum);

glVertex2f(scaled\_entrance\_pos\_x,scaled\_entrance\_pos\_y-0.05f/scalenum);

glEnd();

}

/\*\*

\* Rysowanie kasy w oknie OpenGL

\*/

public void DrawCounter(){

glBegin(GL\_QUADS);

glColor3f(0f,0.5f,0.5f);

glVertex2f(scaled\_counter\_pos\_x,scaled\_counter\_pos\_y);

glVertex2f(scaled\_counter\_pos\_x+0.7f/scalenum,scaled\_counter\_pos\_y);

glVertex2f(scaled\_counter\_pos\_x+0.7f/scalenum,scaled\_counter\_pos\_y+0.2f/scalenum);

glVertex2f(scaled\_counter\_pos\_x,scaled\_counter\_pos\_y+0.2f/scalenum);

glEnd();

}

/\*\*

\* Rysowanie Regałów w oknie OpenGL

\*/

public void DrawShelves(){

for (int i = 0 ; i<scaled\_shelves.size(); ++i){

glBegin(GL\_QUADS);

glColor3fv(DefineProduct(scaled\_shelves.get(i)));

glVertex2f(scaled\_shelves.get(i).shelf\_pos\_x+0.4f/scalenum,scaled\_shelves.get(i).shelf\_pos\_y+0.4f/scalenum);

glVertex2f(scaled\_shelves.get(i).shelf\_pos\_x+0.6f/scalenum,scaled\_shelves.get(i).shelf\_pos\_y+0.4f/scalenum);

glVertex2f(scaled\_shelves.get(i).shelf\_pos\_x+0.6f/scalenum,scaled\_shelves.get(i).shelf\_pos\_y+0.9f/scalenum);

glVertex2f(scaled\_shelves.get(i).shelf\_pos\_x+0.4f/scalenum,scaled\_shelves.get(i).shelf\_pos\_y+0.9f/scalenum);

glEnd();

}

}

/\*\*

\* metoda do definiowania koloru z Enum Product

\* @return float[r,g,b]

\*/

private float[] DefineProduct(Shelf shelf){

if(shelf.product == Product.APPLE){

return new float[]{1f,0f,0f};

}

if(shelf.product == Product.BREAD){

return new float[]{1f,1f,0f};

}

if(shelf.product == Product.CORN){

return new float[]{1f,0.5f,0.1f};

}

if(shelf.product == Product.MEAT){

return new float[]{0.7f,0.2f,0.1f};

}

if(shelf.product == Product.WATER){

return new float[]{0f,0.5f,0.5f};

}

else return new float[]{1f,1f,1f};

}

/\*\*

\* Metoda do nowego losowania koloru ścian i produktów

\*/

public void SetRandomValues(){

wallColor = WallColor.values()[rnd.nextInt(WallColor.values().length)];

for (int i = 0 ; i<scaled\_shelves.size(); ++i){

scaled\_shelves.get(i).product = Product.values()[rnd.nextInt(Product.values().length)];

}

}

}

**Product Enum**

package com.persei;

/\*\*

\* Enum ze spisem produktów

\* \*/

public enum Product {

APPLE,

CORN,

MILK,

MEAT,

BREAD,

WATER

}

**WallColor Enum**

package com.persei;

/\*\*

\* Enum ze spisem kolorów ścian sklepu

\*\*/

public enum WallColor {

Red,

Blue,

Green,

Black,

White

}