Autor:Taras Kuts 16555

Dokumentacja

Techniczna

Klasy:

Apple,AudioPlayer,Button,IDao,DBConnect,Enemy,Fonts,Game,IHelper,Level,

MouseInput,Player,spriteSheet,Texture,Tile,Users,UsersDao

Apple:

**public** **class** Apple **extends** Rectangle{

**public** Apple(**int** x,**int** y) {

setBounds(x+10,y+8,12,12);

}

**public** **void** render(Graphics g) {

g.drawImage(Texture.*apple*,x,y,width,height,**null**);

}

AudioPlayer:

**public** **class** AudioPlayer {

**private** **static** **final** **int** ***BUFFER\_SIZE*** = 4096;//size of the byte buffer used to read/write the audio stream

**public** **static** **void** play(String audioFilePath) {

File audioFile = **new** File(audioFilePath);

**try** {

AudioInputStream audioStream = AudioSystem.*getAudioInputStream*(audioFile);

AudioFormat format = audioStream.getFormat();

DataLine.Info info = **new** DataLine.Info(SourceDataLine.**class**, format);

SourceDataLine audioLine = (SourceDataLine) AudioSystem.*getLine*(info);

audioLine.open(format);

audioLine.start();

System.***out***.println("Playback started.");

**byte**[] bytesBuffer = **new** **byte**[***BUFFER\_SIZE***];

**int** bytesRead = -1;

**while** ((bytesRead = audioStream.read(bytesBuffer)) != -1) {

audioLine.write(bytesBuffer, 0, bytesRead);

}

//Stop\_audio(audioStream,audioLine);

//play("C:\\Users\\grene\\eclipse-workspace\\Pac-Man\\res\\audio\\CantinaBand60.wav");

} **catch** (UnsupportedAudioFileException ex) {

System.***out***.println("The specified audio file is not supported.");

ex.printStackTrace();

} **catch** (LineUnavailableException ex) {

System.***out***.println("Audio line for playing back is unavailable.");

ex.printStackTrace();

} **catch** (IOException ex) {

System.***out***.println("Error playing the audio file.");

ex.printStackTrace();

}

}

**public** **static** **void** Stop\_audio(AudioInputStream audioStream,SourceDataLine audioLine) **throws** IOException {

audioLine.drain();

audioLine.close();

audioStream.close();

}

Button:

**public** **class** Button **extends** Rectangle{

**private** Font font,selectedFont;

**private** Color color,selectedColor;

**private** **boolean** selected;

**private** String text;

**private** **int** textY;

**public** Button(String text,**int** textY,Font font,Font selectedFont,Color color,Color selectedColor) {

**this**.text = text;

**this**.textY = textY;

**this**.font = font;

**this**.selectedFont = selectedFont;

**this**.color = color;

**this**.selectedColor = selectedColor;

}

**public** **void** setSelected(**boolean** selected) {

**this**.selected = selected;

}

**public** **void** render(Graphics g) {

**if**(selected) {

Fonts.*drawString*(g, selectedFont, selectedColor, text,textY);

}

**else** {

Fonts.*drawString*(g, font, color, text,textY);

}

FontMetrics fm = g.getFontMetrics();

**this**.x = (Game.***WIDTH*** - fm.stringWidth(text))/2;

**this**.y = textY - fm.getHeight();

**this**.width = fm.stringWidth(text);

**this**.height = fm.getHeight();

g.drawRect(x, y, width, height);

}

Dao:

**public** **interface** Dao<T> {

**void** save(T t);

**void** update(T t, String[] params);

**void** remove(T t);

}

DBConnect:

**public** **class** DBConnect {

Connection conn;

**public** **static** Connection ConnectDB()

{

**try**{

Class.*forName*("org.sqlite.JDBC");

Connection conn = DriverManager.*getConnection*("jdbc:sqlite:E:\\sqlite\\Players.db");

System.***out***.println("connected to database");

**return** conn;

}

**catch**(Exception e){

JOptionPane.*showMessageDialog*(**null**,e);

**return** **null**;

}

}

Enemy:

**import** java.awt.Graphics;

**import** java.awt.Rectangle;

**import** java.util.Random;

**public** **class** Enemy **extends** Rectangle **implements** IHelper{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** **int** random =0,smart =1,find\_path = 2;

**private** **int** state = smart;

**private** **int** right =0,left=1,up=2,down=3;

**private** **int** dir = -1;

**private** **int** timeForAnim =0;

**public** Random randomGen;

Player player;

**private** **int** time = 0;

**private** **int** targetTime = 250;

**private** **int** targetTimeForAnim = 200;

**private** **int** spd = 4;

**private** **int** lastDir = -1;

**public** **int** imageIndex = 0;

**public** Enemy(**int** x,**int** y) {

randomGen = **new** Random();

setBounds(x,y,32,32);

dir = randomGen.nextInt(4);

}

**public** **void** ForAnimation() {

timeForAnim++;

**if**(timeForAnim == targetTimeForAnim) {

timeForAnim =0;

imageIndex++;

}

}

**private** **boolean** canMove(**int** nextx,**int** nexty) {

ForAnimation();

Rectangle bounds = **new** Rectangle(nextx,nexty,width,height);

Level level = Game.*level*;

**for**(**int** xx=0;xx<level.tiles.length;++xx) {

**for**(**int** yy=0;yy < level.tiles[0].length;++yy) {

**if**(level.tiles[xx][yy] != **null**) {

**if**(bounds.intersects(level.tiles[xx][yy])) {

**return** **false**;

}

}

}

} **return** **true**;

}

@Override

**public** **void** tick() {

// **TODO** Auto-generated method stub

**if**(state == random) {

**if**(dir == right) {

**if**(canMove(x+spd,y)) {

**if**(randomGen.nextInt(100) < 50) x+=spd;

}**else** {

dir = randomGen.nextInt(4);

}

}**else** **if**(dir == left) {

**if**(canMove(x-spd,y)) {

**if**(randomGen.nextInt(100) < 50) x-=spd;

}**else** {

dir = randomGen.nextInt(4);

}

}**else** **if**(dir == up) {

**if**(canMove(x,y-spd)) {

**if**(randomGen.nextInt(100) < 50) y-=spd;

}**else** {

dir = randomGen.nextInt(4);

}

}**else** **if**(dir == down)

{

**if**(canMove(x,y+spd)) {

**if**(randomGen.nextInt(100) < 50) y+=spd;

}**else** {

dir = randomGen.nextInt(4);

}

}

time++;

**if**(time == targetTime) state = smart;

}**else** **if**(state == smart) {

**boolean** move = **false**;

**if**(x < Game.*player*.x) {

**if**(canMove(x+spd,y)) {

**if**(randomGen.nextInt(100) < 50) x+=spd;

move = **true**;

lastDir = right;

}

}

**if**(x > Game.*player*.x) {

**if**(canMove(x-spd,y)) {

**if**(randomGen.nextInt(100) < 50) x-=spd;

move = **true**;

lastDir = left;

}

}

**if**(y < Game.*player*.y) {

**if**(canMove(x,y+spd)) {

**if**(randomGen.nextInt(100) < 50) y+=spd;

move = **true**;

lastDir = down;

}

}

**if**(y > Game.*player*.y) {

**if**(canMove(x,y-spd)) {

**if**(randomGen.nextInt(100) < 50) y-=spd;

move = **true**;

lastDir = up;

}

}

**if**(x == Game.*player*.x && y == Game.*player*.y) move = **true**;

**if**(!move) {

state = find\_path;

}

time++;

**if**(time==targetTime) {

state = random;

time = 0;

}

//Follow the player

}**else** **if**(state == find\_path) {

**if**(lastDir == right) {

**if**(y<Game.*player*.y) {

**if**(canMove(x,y+spd)) {

**if**(randomGen.nextInt(100) < 50) y+=spd;

state = smart;

}

}

**else** {

**if**(canMove(x,y-spd)) {

**if**(randomGen.nextInt(100) < 50) y-=spd;

state = smart;

}

}

**if**(canMove(x+spd,y)) {

**if**(randomGen.nextInt(100) < 50) x+=spd;

}

}**else** **if**(lastDir == left) {

**if**(y<Game.*player*.y) {

**if**(canMove(x,y+spd)) {

**if**(randomGen.nextInt(100) < 50) y+=spd;

state = smart;

}

}

**else** {

**if**(canMove(x,y-spd)) {

**if**(randomGen.nextInt(100) < 50) y-=spd;

state = smart;

}

}

**if**(canMove(x-spd,y)) {

**if**(randomGen.nextInt(100) < 50)x-=spd;

}

}**else** **if**(lastDir == up) {

**if**(x<Game.*player*.y) {

**if**(canMove(x+spd,y)) {

**if**(randomGen.nextInt(100) < 50) x+=spd;

state = smart;

}

}

**else** {

**if**(canMove(x-spd,y)) {

**if**(randomGen.nextInt(100) < 50) x-=spd;

state = smart;

}

}

**if**(canMove(x,y-spd)) {

**if**(randomGen.nextInt(100) < 50) y-=spd;

}

}**else** **if**(lastDir == down) {

**if**(x<Game.*player*.y) {

**if**(canMove(x+spd,y)) {

**if**(randomGen.nextInt(100) < 50) x+=spd;

state = smart;

}

}

**else** {

**if**(canMove(x-spd,y)) {

**if**(randomGen.nextInt(100) < 50) x-=spd;

state = smart;

}

}

**if**(canMove(x,y+spd)) {

**if**(randomGen.nextInt(100) < 50) y+=spd;

}

}

time++;

**if**(time==targetTime) {

state = random;

time = 0;

}

}

}

@Override

**public** **void** render(Graphics g) {

// **TODO** Auto-generated method stub

g.drawImage(Texture.*ghost*[imageIndex%2],x,y,width,height,**null**);

}

}

Fonts:

import java.awt.Color;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

public class Fonts {

public static void drawString(Graphics g,Font f,Color c,String text,int x,int y)

{

g.setColor(c);

g.setFont(f);

g.drawString(text, x, y);

}

public static void drawString(Graphics g,Font f,Color c,String text) {

FontMetrics fm = g.getFontMetrics(f);

int x = (Game.WIDTH - fm.stringWidth(text))/2; //horizontal center

int y = ((Game.HEIGHT - fm.getHeight())/2)+fm.getAscent(); //vertical center

drawString(g,f,c,text,x,y);

}

public static void drawString(Graphics g,Font f,Color c,String text,double x) {

FontMetrics fm = g.getFontMetrics(f);

int y = ((Game.HEIGHT - fm.getHeight())/2)+fm.getAscent(); //vertical center

drawString(g,f,c,text,(int)x,y);

}

public static void drawString(Graphics g,Font f,Color c,String text,int y) {

FontMetrics fm = g.getFontMetrics(f);

int x = (Game.WIDTH - fm.stringWidth(text))/2; //horizontal center

drawString(g,f,c,text,x,y);

}

}

Game:

**import** java.awt.BorderLayout;

**import** java.awt.Canvas;

**import** java.awt.Color;

**import** java.awt.Container;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.Graphics;

**import** java.awt.Rectangle;

**import** java.awt.image.BufferStrategy;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** Game **extends** Canvas **implements** Runnable,KeyListener,IHelper{

**public** **static** JFrame *frame* = **new** JFrame();

**public** **static** JPanel *jp* = **new** JPanel();

**public** **static** JFrame *login* = **new** JFrame();

**public** **static** JFrame *registration* = **new** JFrame();

**public** **static** **final** **int** ***WIDTH*** = 640,***HEIGHT*** = 480;

**public** **static** **final** String ***TITLE*** = "Pac-Man";

**public** **static** **final** String ***TITLE1*** = "Colors of PAC-MAN";

**private** **boolean** isRunning = **false**;

**private** Thread thread;

// Модель списка

**private** **final** String[] items = { "Blue" ,"White" ,"Yellow"};

**private** **final** JList<String> list = **new** JList<String>(items);

**private** **final** JFrame fram = **new** JFrame();

**private** IHelper helper;

**private** Button[] options;//selection 0,1,2

**private** **int** currentSelection;

**private** **static** AudioPlayer *Audioplayer*;

**public** **static** Player *player*;

**public** **static** Level *level*;

**public** **static** spriteSheet *spritesheet*;

**public** **static** **final** **int** ***PAUSE\_SCREEN*** = 0,***GAME*** = 1,***END*** = 2,***WIN*** =3;

**public** **static** **int** *STATE* = -1;

**public** **boolean** isEnter = **false**;

**private** **int** time = 0;

**private** **int** targetFrames = 15;

**private** **boolean** showText = **true**;

**private** ImageIcon background = **new** ImageIcon("C:/Users/grene/eclipse-workspace/Pac-Man/res/Images/background.jpg");

**private** **static** Container *containerLogIn*;

**private** **static** JLabel *titleLogIn*;

**private** **static** JLabel *nameLogIn*;

**private** **static** JLabel *passwordLogIn*;

**private** **static** JTextField *tnameLogIn*;

**private** **static** JButton *LogIN*;

**private** **static** JButton *registr*;

**public** **static** String *UserNameLogIn*;

**public** **static** String *PasswordLogIn*[];

**private** **static** Users *UsersLogLogIn*;

**private** **static** Container *containerRegister*;

**private** **static** JLabel *titleRegister*;

**private** **static** JLabel *nameRegister*;

**private** **static** JLabel *passwordRegister*;

**private** **static** JTextField *tpasswordRegister*;

**private** **static** JTextField *tnameRegister*;

**private** **static** JPasswordField *pass*;

**private** **static** JButton *RegistationBTN*;

**public** **static** String *UserNameRegister*;

**public** **static** String *PasswordRegister*;

**public** **static** String *email*;

**public** **static** **void** main(String[] args)

{

*Login*();

}

@Override

**public** **void** keyPressed(KeyEvent e) {

// **TODO** Auto-generated method stub

**if**(*STATE*==***GAME***) {

**if**(e.getKeyCode() == KeyEvent.***VK\_RIGHT***) *player*.right = **true**;

**if**(e.getKeyCode() == KeyEvent.***VK\_LEFT***) *player*.left = **true**;

**if**(e.getKeyCode() == KeyEvent.***VK\_UP***) *player*.up = **true**;

**if**(e.getKeyCode() == KeyEvent.***VK\_DOWN***) *player*.down = **true**;

}**else** **if**(*STATE* == Game.***PAUSE\_SCREEN***) {

**if**(e.getKeyCode() == KeyEvent.***VK\_ENTER***) {

isEnter = **true**;

}

**if**(e.getKeyCode() == KeyEvent.***VK\_Q***) {

System.*exit*(1);

}

}

}

@Override

**public** **void** keyReleased(KeyEvent e) {

**if**(e.getKeyCode() == KeyEvent.***VK\_RIGHT***) *player*.right = **false**;

**if**(e.getKeyCode() == KeyEvent.***VK\_LEFT***) *player*.left = **false**;

**if**(e.getKeyCode() == KeyEvent.***VK\_UP***) *player*.up = **false**;

**if**(e.getKeyCode() == KeyEvent.***VK\_DOWN***) *player*.down = **false**;

}

@Override

**public** **void** keyTyped(KeyEvent e) {

// **TODO** Auto-generated method stub

}

**public** Game()

{

Dimension dimension = **new** Dimension(Game.***WIDTH***,Game.***HEIGHT***);

setPreferredSize(dimension);

setMinimumSize(dimension);

setMaximumSize(dimension);

addKeyListener(**this**);

MouseInput mi = **new** MouseInput();

addMouseListener(mi);

addMouseMotionListener(mi);

*STATE* = ***PAUSE\_SCREEN***;

Thread thread = **new** Thread(() ->*Audioplayer*.*play*("C:\\Users\\grene\\eclipse-workspace\\Pac-Man\\res\\audio\\CantinaBand60.wav"));//tworzymy watek

thread.start();

ButtonsForMenu();

}

**public** **void** ButtonsForMenu() {

options = **new** Button[5];

options[0] = **new** Button("Easy level",200 + 0\*60,

**new** Font("Arial",Font.***PLAIN***,32),**new** Font("Arial",Font.***BOLD***,42),

Color.***WHITE***,Color.***YELLOW***);

options[1] = **new** Button("Medium level",200 + 1\*60,

**new** Font("Arial",Font.***PLAIN***,32),**new** Font("Arial",Font.***BOLD***,42),

Color.***WHITE***,Color.***YELLOW***);

options[2] = **new** Button("Hard level",200 + 2\*60,

**new** Font("Arial",Font.***PLAIN***,32),**new** Font("Arial",Font.***BOLD***,42),

Color.***WHITE***,Color.***YELLOW***);

options[3] = **new** Button("Very Hard level",200 + 3\*60,

**new** Font("Arial",Font.***PLAIN***,32),**new** Font("Arial",Font.***BOLD***,42),

Color.***WHITE***,Color.***YELLOW***);

options[4] = **new** Button("Exit",200 + 4\*60,

**new** Font("Arial",Font.***PLAIN***,32),**new** Font("Arial",Font.***BOLD***,42),

Color.***WHITE***,Color.***YELLOW***);

}

**public** **synchronized** **void** Stop()

{

**if**(!isRunning) **return**;

isRunning = **false**;

**try** {

thread.join();

} **catch** (InterruptedException e) {

//

e.printStackTrace();

}

}

**public** **synchronized** **void** Start()

{

**if**(isRunning) **return**;

isRunning = **true**;

thread = **new** Thread(**this**);

thread.start();

}

**public** **void** SkinLVL(String map,String skins) {

isEnter = **false**;

*player* = **new** Player(Game.***WIDTH***/2,Game.***HEIGHT***/2);

*level* = **new** Level(map);

*spritesheet* = **new** spriteSheet(skins);

**new** Texture();

*STATE* = ***GAME***;

}

**public** **void** Blinking() {

time++;

**if**(time == targetFrames) {

time =0;

**if**(showText) {

showText = **false**;

}**else** {

showText = **true**;

}

}

}

**private** **void** rendering()

{

BufferStrategy bs = getBufferStrategy();

**if**(bs == **null**) {

createBufferStrategy(3);

**return**;

}

Graphics g = bs.getDrawGraphics();

g.drawImage(background.getImage(),0,0,**null**);

g.fillRect(0, 0, Game.***WIDTH***, Game.***HEIGHT***);

**if**(*STATE* == ***GAME***){

*player*.render(g);

*level*.render(g);

}**else** **if**(*STATE* == ***PAUSE\_SCREEN***) {

**int** boxWidth = 640;

**int** boxHeight = 480;

**int** xx = Game.***WIDTH*** / 2;

**int** yy = Game.***HEIGHT*** /2;

g.fillRect(xx, yy, boxWidth, boxHeight);

render(g);

g.setFont(**new** Font(Font.***DIALOG***,Font.***BOLD***,18));

**if**(showText)g.drawString("Autor:Taras Kuts", xx+150, yy+240);

}

g.dispose();

bs.show();

}

@Override

**public** **void** run() {

requestFocus();

**int** fps = 0;

**double** timer = System.*currentTimeMillis*();

**long** LastTime = System.*nanoTime*();

**double** targetTick = 60.0;

**double** delta = 0;

**double** ns = 1000000000/targetTick;

**while**(isRunning)

{

**long** now = System.*nanoTime*();

delta+=(now - LastTime)/ns;

LastTime = now;

**while**(delta >= 1)

{

ticki();

tick();

MouseInput.*update*();

rendering();

fps++;

delta--;

}

**if**(System.*currentTimeMillis*() - timer >= 1000){

System.***out***.println(fps);

fps = 0;

timer+=1000;

}

}

Stop();

}

**public** **void** POM(String case1,String case2,String case3,String caseMap) {

fram.setPreferredSize(**new** Dimension(200

, 200));

fram.add(list, BorderLayout.***CENTER***);

fram.setResizable(**false**);

fram.pack();

fram.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

fram.setLocationRelativeTo(**null**);

fram.setTitle(Game.***TITLE1***);

fram.add(**new** JLabel("Choose color of Pac-Man"), BorderLayout.***EAST***);

JButton btn = **new** JButton("Choose");

fram.add(btn, BorderLayout.***SOUTH***);

btn.setPreferredSize(**new** Dimension(30, 30));

btn .setFont(**new** Font("Arial", Font.***PLAIN***, 40));

btn.setBackground(Color.***GREEN***);

btn.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

JOptionPane.*showConfirmDialog*(fram, "You Selected : " + list.getSelectedValue(), "Display",

JOptionPane.***PLAIN\_MESSAGE***);

System.***out***.println("i" + currentSelection);

**if**(list.getSelectedIndex() == 0 ) {

System.***out***.println("i" + currentSelection);

fram.setVisible(**false**);

**if**(currentSelection == 0) {

SkinLVL("/map/map1.png","s/s2blue1.png");

}

**if**(currentSelection == 1) {

SkinLVL("/map/map2.png","s/s2blue1.png");

}

**if**(currentSelection == 2) {

SkinLVL("/map/map3.png","s/s2blue1.png");

}

**if**(currentSelection == 3) {

SkinLVL("/map/map4.png","s/s2blue1.png");

}

//SkinLVL(caseMap,case1);

*STATE* = ***GAME***;

}

**else** **if**(list.getSelectedIndex()==1) {

System.***out***.println("i" + currentSelection);

**if**(currentSelection == 0) {

SkinLVL("/map/map1.png","s/s2white1.png");

}

**if**(currentSelection == 1) {

SkinLVL("/map/map2.png","s/s2white1.png");

}

**if**(currentSelection == 2) {

SkinLVL("/map/map3.png","s/s2white1.png");

}

**if**(currentSelection == 3) {

SkinLVL("/map/map4.png","s/s2white1.png");

}

fram.setVisible(**false**);

//SkinLVL(caseMap,case2);

*STATE* = ***GAME***;

}

**else** **if**(list.getSelectedIndex()==2){

System.***out***.println("i" + currentSelection);

**if**(currentSelection == 0) {

SkinLVL("/map/map1.png","s/s2.png");

}

**if**(currentSelection == 1) {

SkinLVL("/map/map2.png","s/s2.png");

}

**if**(currentSelection == 2) {

SkinLVL("/map/map3.png","s/s2.png");

}

**if**(currentSelection == 3) {

SkinLVL("/map/map4.png","s/s2.png");

}

fram.setVisible(**false**);

//SkinLVL(caseMap,case3);

*STATE* = ***GAME***;

}

**else** {

JOptionPane.*showConfirmDialog*(fram, "Selected one color : " ,"Display",

JOptionPane.***PLAIN\_MESSAGE***);

}

}

});

fram.setVisible(**true**);

}

**public** **static** **void** Login() {

*login*.setPreferredSize(**new** Dimension(480

, 400));

*login*.setResizable(**false**);

*login*.pack();

*login*.setLocationRelativeTo(**null**);

*login*.setTitle("Login Form");

*login*.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

*login*.setResizable(**false**);

*containerLogIn* = *login*.getContentPane();

*containerLogIn*.setLayout(**null**);

*titleLogIn* = **new** JLabel("Login");

*titleLogIn*.setFont(**new** Font("Arial", Font.***PLAIN***, 56));

*titleLogIn*.setSize(200, 200);

*titleLogIn*.setLocation(175, -75);

*containerLogIn*.add(*titleLogIn*);

*nameLogIn* = **new** JLabel("Username");

*nameLogIn*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*nameLogIn*.setSize(120, 50);

*nameLogIn*.setLocation(10, 105);

*containerLogIn*.add(*nameLogIn*);

*tnameLogIn* = **new** JTextField();

*tnameLogIn*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*tnameLogIn*.setSize(275, 30);

*tnameLogIn*.setLocation(140, 115);

*containerLogIn*.add(*tnameLogIn*);

*passwordLogIn* = **new** JLabel("Passsword");

*passwordLogIn*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*passwordLogIn*.setSize(275, 50);

*passwordLogIn*.setLocation(10, 150);

*containerLogIn*.add(*passwordLogIn*);

*pass* = **new** JPasswordField();

*pass*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*pass*.setSize(275, 30);

*pass*.setLocation(140, 160);

*containerLogIn*.add(*pass*);

*LogIN* = **new** JButton("Login");

*LogIN*.setBackground(Color.***yellow***);

*LogIN*.setFont(**new** Font("Arial", Font.***PLAIN***, 48));

*LogIN*.setSize(200, 100);

*LogIN*.setLocation(10, 250);

*containerLogIn*.add(*LogIN*);

*registr* = **new** JButton("Registr");

*registr*.setBackground(Color.***GREEN***);

*registr*.setFont(**new** Font("Arial", Font.***PLAIN***, 48));

*registr*.setSize(200, 100);

*registr*.setLocation(270, 250);

*containerLogIn*.add(*registr*);

*login*.setVisible(**true**);

*LogIN*.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

*UserNameLogIn* = *tnameLogIn*.getText();

String myPass=String.*valueOf*(*pass*.getPassword());

UsersDao empDao = **new** UsersDao();

*login*.setVisible(**false**);

*UsersLogLogIn* = empDao.get(*UserNameLogIn*, myPass);

empDao.ConnectionClose();

**if**(*UsersLogLogIn* != **null**)

{

Game game = **new** Game();

*frame*.setTitle(Game.***TITLE***);

*frame*.add(game);

*frame*.setResizable(**false**);

*frame*.pack();

*frame*.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

*frame*.setLocationRelativeTo(**null**);

*frame*.setVisible(**true**);

game.Start();

}

**else**{

*login*.setVisible(**true**);

}

}

});

*registr*.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

*Registr*();

*login*.setVisible(**false**);

}

});

}

**public** **static** **void** Registr() {

*registration*.setPreferredSize(**new** Dimension(480

, 400));

*registration*.setResizable(**false**);

*registration*.pack();

*registration*.setLocationRelativeTo(**null**);

*registration*.setTitle("Registration Form");

*registration*.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

*registration*.setResizable(**false**);

*containerRegister* = *registration*.getContentPane();

*containerRegister*.setLayout(**null**);

*titleRegister* = **new** JLabel("Registration");

*titleRegister*.setFont(**new** Font("Arial", Font.***PLAIN***, 42));

*titleRegister*.setSize(300, 550);

*titleRegister*.setLocation(130, -250);

*containerRegister*.add(*titleRegister*);

*nameRegister* = **new** JLabel("Username");

*nameRegister*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*nameRegister*.setSize(120, 50);

*nameRegister*.setLocation(10, 105);

*containerRegister*.add(*nameRegister*);

*tnameRegister* = **new** JTextField();

*tnameRegister*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*tnameRegister*.setSize(275, 30);

*tnameRegister*.setLocation(140, 115);

*containerRegister*.add(*tnameRegister*);

*passwordRegister* = **new** JLabel("Passsword");

*passwordRegister*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*passwordRegister*.setSize(275, 50);

*passwordRegister*.setLocation(10, 150);

*containerRegister*.add(*passwordRegister*);

*tpasswordRegister* = **new** JTextField();

*tpasswordRegister*.setFont(**new** Font("Arial", Font.***PLAIN***, 24));

*tpasswordRegister*.setSize(275, 30);

*tpasswordRegister*.setLocation(140, 160);

*containerRegister*.add(*tpasswordRegister*);

*RegistationBTN* = **new** JButton("Registration");

*RegistationBTN*.setBackground(Color.***green***);

*RegistationBTN*.setFont(**new** Font("Arial", Font.***PLAIN***, 48));

*RegistationBTN*.setSize(460, 100);

*RegistationBTN*.setLocation(5, 250);

*containerRegister*.add(*RegistationBTN*);

*registration*.setVisible(**true**);

*RegistationBTN*.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

*UserNameRegister* = *tnameRegister*.getText();

*PasswordRegister* = *tpasswordRegister*.getText();

**if**(*tnameRegister*.getText().isEmpty() || *tpasswordRegister*.getText().isEmpty())

{

JOptionPane.*showMessageDialog*(**null**, "username or password is empty");

}

**else** **if**(*UserNameRegister*.length() <6 || *PasswordRegister*.length() < 6) {

JOptionPane.*showMessageDialog*(**null**, "username and password must have minimum 6 chars");

}

**else** {

*pass*.setText("");

*tnameLogIn*.setText("");

Users user = **new** Users(*UserNameRegister*,*PasswordRegister*);

UsersDao dao = **new** UsersDao();

dao.save(user);

*registration*.setVisible(**false**);

*Login*();

}

}

});

}

**public** **void** ticki() {

currentSelection = 0;

**boolean** clicked = **false**;

**for**(**int** i=0;i<options.length;++i) {

**if**(options[i].intersects(**new** Rectangle(MouseInput.*getX*(),MouseInput.*getY*(),1,1))){

currentSelection =i;

clicked = MouseInput.*wasPressed*(MouseEvent.***BUTTON1***);

}

}

**if**(clicked )

{

select();

}

}

**private** **void** select() {

**switch**(currentSelection) {

**case** 0:

POM("/s/s2blue1.png","/s/s2white1.png","/s/s2.png","/map/map1.png");

//System.out.println("0");

**break**;

**case** 1:

POM("/s/s2blue1.png","/s/s2white1.png","/s/s2.png","/map/map2.png");

//System.out.println("1");

**break**;

**case** 2:

POM("/s/s2blue1.png","/s/s2white1.png","/s/s2.png","/map/map3.png");

//System.out.println("2");

**break**;

**case** 3:

POM("/s/s2blue1.png","/s/s2white1.png","/s/s2.png","/map/map4.png");

//System.out.println("3");

**break**;

**case** 4:

System.*exit*(1);

**break**;

}

}

@Override

**public** **void** tick() {

// **TODO** Auto-generated method stub

**if**(*STATE* == ***GAME***) {

ButtonsForMenu();

*player*.tick();

*level*.tick();

}**else** **if**(*STATE* == ***PAUSE\_SCREEN***) {

Blinking();

}

**if**(*STATE* == ***WIN***) {

**int** op = JOptionPane.*showConfirmDialog*(**this**, "You are passed this level!Do you want comeback to main menu?.", "Congratulation!", JOptionPane.***OK\_OPTION***);

**if** (op == JOptionPane.***YES\_OPTION***) {

*STATE* = ***PAUSE\_SCREEN***;

}**else** {System.*exit*(1);}

}

}

@Override

**public** **void** render(Graphics g) {

// **TODO** Auto-generated method stub

g.fillRect(0,0,Game.***WIDTH***,Game.***HEIGHT***);

Fonts.*drawString*(g,**new** Font("Arial",Font.***BOLD***,72),Color.***ORANGE***,Game.***TITLE***,80);

**for**(**int** i=0;i<options.length;++i) {

**if**(i== currentSelection)

options[i].setSelected(**true**);

**else** options[i].setSelected(**false**);

options[i].render(g);

}

}

}

Ihelper:

**public** **interface** IHelper {

**public** **void** tick();

**public** **void** render(Graphics g);

}

Level:

import java.awt.Graphics;

import java.util.List;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.util.ArrayList;

import javax.imageio.ImageIO;

public class Level implements IHelper{

public int width;

public int height;

public Tile[][] tiles;

public List<Apple> apples;

public List<Enemy> enemies;

public Level(String path)

{

apples = new ArrayList<>();

enemies = new ArrayList<>();

try {

BufferedImage map = ImageIO.read(getClass().getResource(path));

this.width = map.getWidth();

this.height = map.getHeight();

int[] pixels = new int[width\*height];

tiles = new Tile[width][height];

map.getRGB(0, 0,width,height,pixels,0,width);

for(int xx=0;xx < width;++xx) {

for(int yy=0;yy<height;++yy) {

int val = pixels[xx + (yy\*width)];

if(val == 0xFF000000) {

//TILE

tiles[xx][yy] = new Tile(xx\*32,yy\*32);

}else if(val == 0xFF0000FF) {

//player

Game.player.x = xx\*32;

Game.player.y = yy\*32;

}else if(val == 0xFFFF0000) {

//enemy

enemies.add(new Enemy(xx\*32,yy\*32));

}else {

apples.add(new Apple(xx\*32,yy\*32));

}

}

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public void tick() {

// TODO Auto-generated method stub

for (int i=0;i<enemies.size();++i)

{

enemies.get(i).tick();

}

}

@Override

public void render(Graphics g) {

// TODO Auto-generated method stub

for(int x=0;x<width;++x) {

for(int y=0;y<height;++y) {

if(tiles[x][y] != null) tiles[x][y].render(g);

}

}

for(int i=0;i<apples.size();++i)

{

apples.get(i).render(g);

}

for (int i=0;i<enemies.size();++i)

{

enemies.get(i).render(g);

}

}

}

MouseInput:

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class MouseInput extends MouseAdapter{

private static final int NUM\_BUTTONS = 10;

private static final boolean[] buttons = new boolean[NUM\_BUTTONS];

private static final boolean[] lastButtons = new boolean[NUM\_BUTTONS];

private static int x = -1,y = -1;

private static int lastX = x,lastY = y;

private static boolean moving;

@Override

public void mousePressed(MouseEvent e) {

//System.out.println("Button:" + e.getButton());

buttons[e.getButton()] = true;

}

@Override

public void mouseReleased(MouseEvent e) {

buttons[e.getButton()] = false;

}

@Override

public void mouseMoved(MouseEvent e) {

x = e.getX();

y = e.getY();

moving = true;

}

public static void update() {

for(int i=0;i<NUM\_BUTTONS;++i) {

lastButtons[i] = buttons[i];

//TODO: Do we need to check for how long the mouse has been still?only set

//moving to false when the mouse has been static for 5 seconds

if(x == lastX && y == lastY) moving = false;

lastX = x;

lastY = y;

}

}

public static boolean isDown(int button){

return buttons[button];

}

public static boolean wasPressed(int button) {

return isDown(button) && !lastButtons[button];

}

public static boolean wasReleased(int button) {

return !isDown(button) && lastButtons[button];

}

public static int getX() {

return x;

}

public static int getY() {

return y;

}

public static boolean isMooving() {

return moving;

}

}

Player:

**import** java.awt.Graphics;

**import** java.awt.Rectangle;

**public** **class** Player **extends** Rectangle **implements** IHelper{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **int** speed = 4;

**private** **int** time=0,targetTime = 30;

**public** **int** imageIndex = 0;

**private** **int** lastDir = 1;

**public** **int** apples = 0;

**public** Player(**int** x,**int** y)

{

setBounds(x,y,32,32);

}

**public** **boolean** right,left,up,down;

**public** **void** ForAnimation() {

time++;

**if**(time == targetTime) {

time =0;

imageIndex++;

}

}

**private** **boolean** canMove(**int** nextx,**int** nexty) {

Rectangle bounds = **new** Rectangle(nextx,nexty,width,height);

Level level = Game.*level*;

**for**(**int** xx=0;xx<level.tiles.length;++xx) {

**for**(**int** yy=0;yy < level.tiles[0].length;++yy) {

**if**(level.tiles[xx][yy] != **null**) {

**if**(bounds.intersects(level.tiles[xx][yy])) {

**return** **false**;

}

}

}

}

**return** **true**;

}

@Override

**public** **void** tick() {

// **TODO** Auto-generated method stub

**if**(right && canMove(x+speed,y)) {

x+=speed;

lastDir=1;

}

**if**(left && canMove(x-speed,y))

{

x-=speed;

lastDir=-1;

}

**if**(up && canMove(x,y-speed))

{

y-=speed;

lastDir = 2;

}

**if**(down && canMove(x,y+speed))

{

y+=speed;

lastDir = -2;

}

Level level = Game.*level*;

**for**(**int** i=0;i<level.apples.size();++i) {

**if**(**this**.intersects(level.apples.get(i))) {

level.apples.remove(i);

**break**;

}

}

**if**(level.apples.size() == 0) {

//Game end,we win!

Game.*STATE* = Game.***WIN***;

**return**;

}

**for**(**int** i = 0;i< Game.*level*.enemies.size();++i)

{

**if**(**this**.intersects(Game.*level*.enemies.get(i))) {

//Menu system

//DevelopersLogo logo = new DevelopersLogo();

Game.*STATE* = Game.***PAUSE\_SCREEN***;

apples=0;

}

ForAnimation();

}

}

@Override

**public** **void** render(Graphics g) {

// **TODO** Auto-generated method stub

**if**(lastDir == 1)

{

g.drawImage(Texture.*player*[imageIndex%2],x,y,width,height,**null**);

}

**else** **if**(lastDir == -1){

g.drawImage(Texture.*player*[imageIndex%2],x+32,y,-width,height,**null**);

}

**else** **if**(lastDir == 2) {

g.drawImage(Texture.*playerUP*[imageIndex%2],x,y,width,height,**null**);

}

**else** {

g.drawImage(Texture.*playerDOWN*[imageIndex % 2], x, y, width, height, **null**);

}

}

}

SpriteSheet:

import java.awt.image.BufferedImage;

import java.io.IOException;

import javax.imageio.ImageIO;

public class spriteSheet {

private BufferedImage sheet;

public spriteSheet(String path)

{

try {

sheet = ImageIO.read(getClass().getResource(path));

}catch(IOException e) {

System.out.println("failed to load image");

}

}

public BufferedImage getSprite(int xx,int yy) {

return sheet.getSubimage(xx, yy,16,16);

}

}

Texture:

**import** java.awt.image.BufferedImage;

**public** **class** Texture {

**public** **static** BufferedImage[] *player*;

**public** **static** BufferedImage[] *ghost*;

**public** **static** BufferedImage *apple*;

**public** **static** BufferedImage[] *playerDOWN*;

**public** **static** BufferedImage[] *playerUP*;

**public** Texture() {

*player* = **new** BufferedImage[2];

*playerDOWN* = **new** BufferedImage[2];

*playerUP* = **new** BufferedImage[2];

*ghost* = **new** BufferedImage[2];

*apple* = Game.*spritesheet*.getSprite(0, 32);

*playerUP*[0] = Game.*spritesheet*.getSprite(64, 0);

*playerUP*[1] = Game.*spritesheet*.getSprite(80, 0);

*playerDOWN*[0] = Game.*spritesheet*.getSprite(48,0);

*playerDOWN*[1] = Game.*spritesheet*.getSprite(32,0);

*player*[0] = Game.*spritesheet*.getSprite(0,0);

*player*[1]= Game.*spritesheet*.getSprite(16,0);

*ghost*[0] = Game.*spritesheet*.getSprite(0, 16);

*ghost*[1] = Game.*spritesheet*.getSprite(16,16);

}

}

Tile:

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Rectangle;

public class Tile extends Rectangle implements IHelper{

public Tile(int x,int y) {

setBounds(x,y,25,25);

}

@Override

public void tick() {

// TODO Auto-generated method stub

}

@Override

public void render(Graphics g) {

// TODO Auto-generated method stub

g.setColor(new Color(52, 116, 128));

g.fillRect(x, y, width, height);

}

}

Users:

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Rectangle;

public class Tile extends Rectangle implements IHelper{

public Tile(int x,int y) {

setBounds(x,y,25,25);

}

@Override

public void tick() {

// TODO Auto-generated method stub

}

@Override

public void render(Graphics g) {

// TODO Auto-generated method stub

g.setColor(new Color(52, 116, 128));

g.fillRect(x, y, width, height);

}

}

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** javax.swing.JOptionPane;

**public** **class** UsersDao **implements** IDao<Users> {

Connection conn;

ResultSet rs;

PreparedStatement pst;

**public** UsersDao() {

conn = DBConnect.*ConnectDB*();

}

**public** **void** ConnectionClose()

{

**try** {

conn.close();

} **catch** (SQLException ex) {

JOptionPane.*showMessageDialog*(**null**, ex);

}

}

**public** Users get(String UserName, String Password)

{

String query = "SELECT \* FROM Players WHERE name = ? AND password = ?";

**try**{

pst = conn.prepareStatement(query);

pst.setString(1, UserName);

pst.setString(2, Password);

rs = pst.executeQuery();

**if**(rs.next()){

**try**

{

Users getUsers = **new** Users(

rs.getString("Password"), rs.getString("Name"));

**return** getUsers;

}**catch**(Exception e)

{

JOptionPane.*showMessageDialog*(**null**, e);

}

}

**else**{

JOptionPane.*showMessageDialog*(**null**, "Incorrent username/password");

}

}**catch**(Exception e)

{

JOptionPane.*showMessageDialog*(**null**, e);

}

**return** **null**;

}

@Override

**public** **void** save(Users t) {

**try**{

String query = "INSERT INTO Players (name, password) VALUES (?,?)";

pst = conn.prepareStatement(query);

pst.setString(1, t.getUsername());

pst.setString(2, t.getPassword());

pst.execute();

JOptionPane.*showMessageDialog*(**null**, "New customer addeed");

}**catch**(Exception e){

JOptionPane.*showMessageDialog*(**null**,e.toString());

}

}

@Override

**public** **void** update(Users t, String[] params) {

}

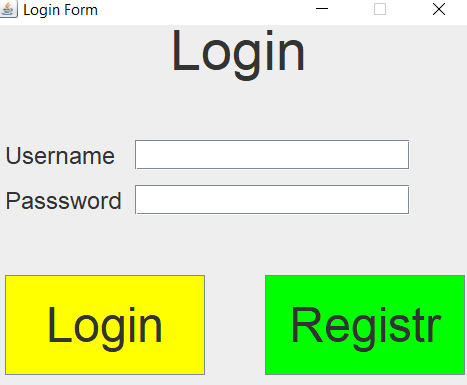
@Override

**public** **void** remove(Users t) {

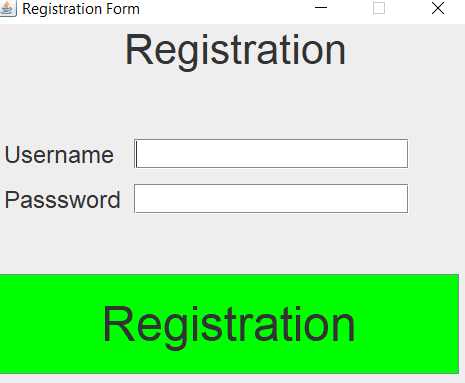
}

}

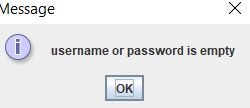
GUI dokumentacja:

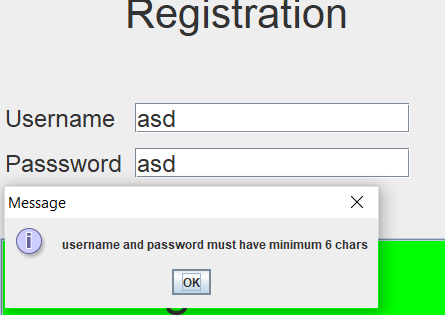


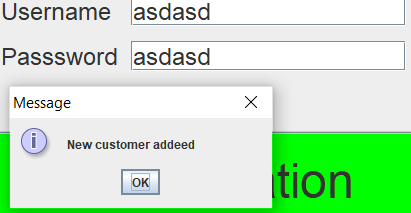
Jezeli uzytkownik nie jest zarejestrowany musi zarejestrowac sie przyciskiem Registr



Pole musi zawierac minimalnie 6 znakow

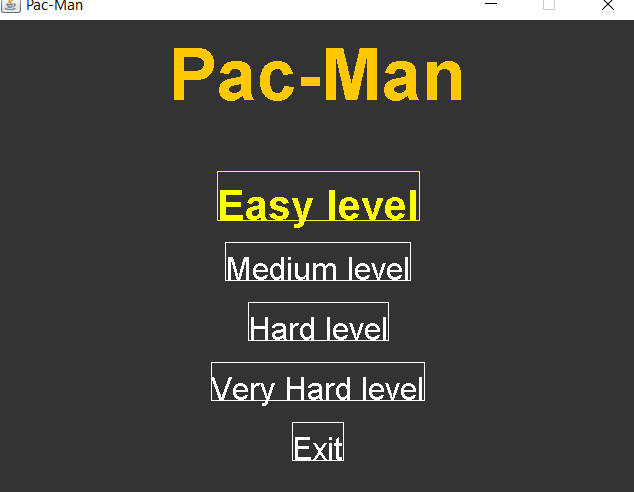






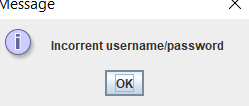
Jezeli prawidlowo byli wpisane dane czyli minimalnie 6 znakow taki uzytkownik bedzie dodany do bazy danych

Jezeli uzytkownik jest poprawny zalogowany otzymuje menu gry i automatycznie zaczyna sie grac muzyka jaka jest powtorzalna az do zamkniecia programu

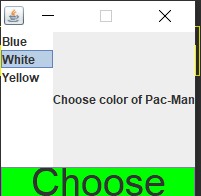


Gdzie moge wybrac sobie poziom skladnosci

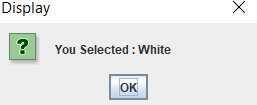
Jezeli nie jest prawidlowy wpisany login lub haslo uzytkownik otrzymuje komunikat po nacisnieciu OK zwraca go na Login do wpisu Logina i Hasla znowu.

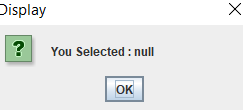


Po nacisnieciu jakogos pozioma skladnosci uzytkownik otzymuje nastepny komunikat

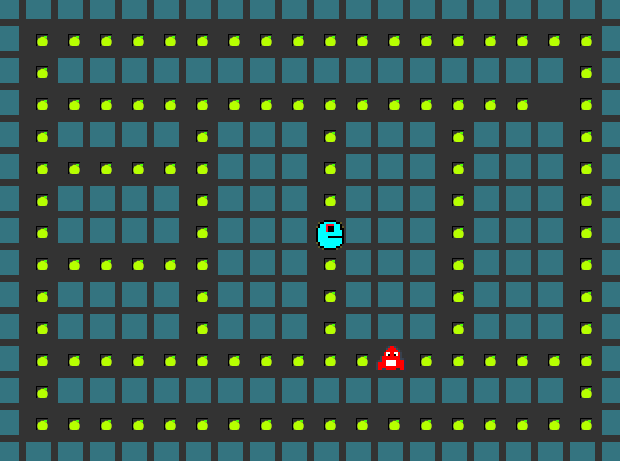


Gdzie wybiera sobie kolor pac-mana(kolor enemyPac-mana tez sie zmienia i frukty do zbierania,mapa)



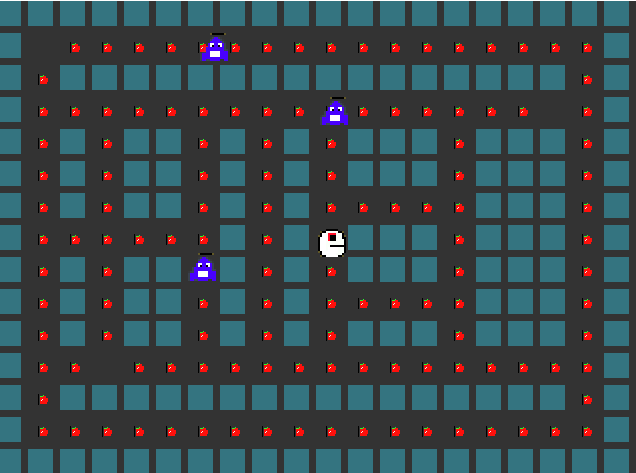
Otrzymuje taki komunikat po wybraniu ,jezeli nic nie bylo wybrano otrzymuje odpowiedni komunikat: 

Tak wyglada pierwszy poziom skladnosci:



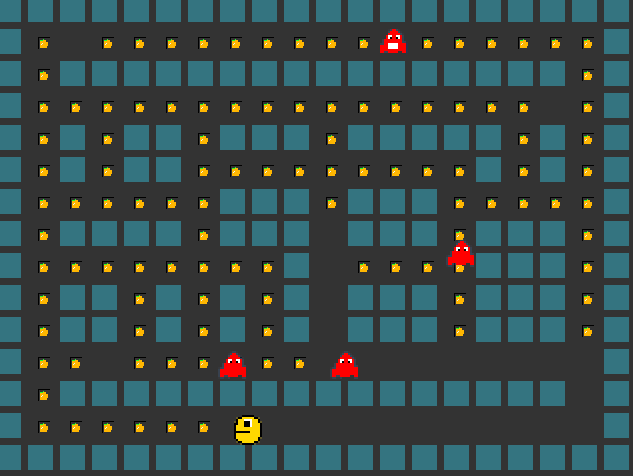
Jeden enemyPac-Man i taka mapa,color byl wybrany niebeski.

Zaraz wybieramy drugi poziom skladnosci i bierzemy teraz bialy kolor Pac-Mana



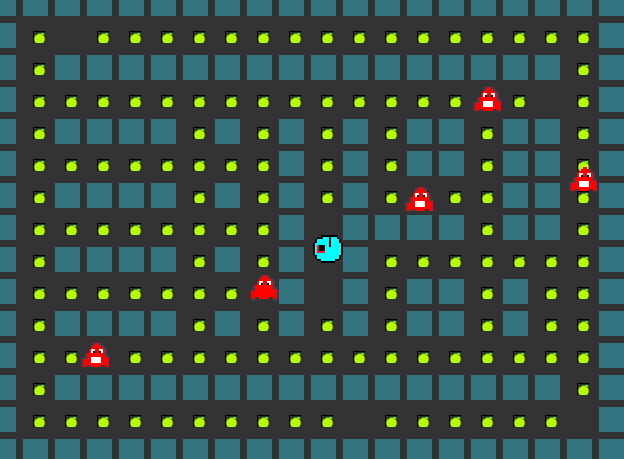
Widzimy ze mapa jest zmieniona,kolor Pac-Mana i enemyPac-Mana tez oprocz tego frukt tez jest zmieniony,ilosc enemyPac-Mana tez jest zmieniona.

Teraz wybieramy trzecij poziom i ostatni kolor jaki jest dostepny to jest zolty:



Kolor zmieniony ilosc enemyPac-Mana tez, szybkosc ruchu Pac-Mana i enemyPac-Mana jest zwiekszona co nastepny poziom czyli na pierwszym wszystko odbywasie wolno a co dalej tym szybszej chodza i enemy i Pac-Mana

I zostal nam ostatni poziom:



Jezeli nasza proba zostala okonczona sukcesem my otzymujemy mastepny komunikat:

