Universitatea Tehnică de Construcții București
Facultatea de Hidrotehnica

Proiect la PCLP III

Profesori Coordonatori:

Sandu Ana-Maria

Oana Flangea

Student:

Beșciu Cristian-Darius

Cuprins

1. Introducere

1.1. Motivarea alegerii temei

Am decis să creez o versiune a jocului Flappy Bird deoarece acesta reprezintă un exemplu potrivit de îmbinare între simplitatea mecanicii și complexitatea implementării. Deși jocul este ușor de înțeles, realizarea sa necesită aplicarea unor concepte importante de programare, precum bucla principală de joc, gestionarea coliziunilor, controlul inputului utilizatorului și organizarea codului într-o manieră orientată pe obiect în Java. Flappy Bird a devenit rapid un joc emblematic, care a marcat o perioadă importantă în istoria jocurilor mobile, iar pentru mulți oameni acesta trezește un sentiment de nostalgie legat de primele experiențe cu jocuri simple, dar dificile, fapt ce a influențat la decizia de a alege acest joc ca și temă.

1.2 Despre Flappy Bird

Flappy Bird este un joc video de tip arcade care a cunoscut o popularitate explozivă în perioada 2013–2014, odată cu răspândirea sa pe platformele mobile iOS și Android. Deși a fost lansat inițial fără o campanie de promovare semnificativă, jocul a atras rapid atenția utilizatorilor datorită mecanicii sale simple și dificultății ridicate. Grafica minimalistă, inspirată din jocurile retro, a contribuit la identitatea sa distinctă și la recunoașterea imediată în rândul jucătorilor.

Conceptul jocului a fost dezvoltat de programatorul vietnamez Dong Nguyen, fondatorul studioului independent dotGears, ca un proiect individual. Într-o perioadă în care

piața jocurilor mobile era dominată de titluri complexe și intens monetizate, Flappy Bird a ieșit în evidență prin simplitatea sa, demonstrând că o idee aparent banală, dar bine executată, poate avea un impact major. Jocul se baza pe un control intuitiv și pe o dificultate intenționat ridicată, menită să provoace perseverența jucătorului.

Succesul jocului a devenit evident la începutul anului 2014, când Flappy Bird a ajuns în topul descărcărilor la nivel global. Acesta a devenit rapid un subiect frecvent pe rețelele sociale, fiind asociat atât cu frustrarea, cât și cu dorința de autodepășire. În mod surprinzător, la apogeul popularității sale, creatorul a decis să retragă jocul din magazinele online, invocând motive legate de impactul negativ asupra utilizatorilor. Această decizie a contribuit la consolidarea statutului său de fenomen cultural.

Astăzi, Flappy Bird este considerat un simbol al erei timpurii a jocurilor mobile și un exemplu reprezentativ al succesului viral. Deși jocul original nu mai este disponibil oficial, el este frecvent recreat și reinterpretat, atât ca exercițiu educațional în programare, cât și ca omagiu adus unei perioade în care simplitatea și creativitatea puteau genera un impact global. Pentru mulți utilizatori, Flappy Bird rămâne un joc asociat cu nostalgia primelor experiențe frustrante, dar memorabile, din lumea aplicațiilor mobile.

2. Concepte și tehnologii folosite în elaborarea proiectului

2.1. Visual Studio Code

Visual Studio Code

Visual Studio Code (prescurtat VS Code) este un editor de cod sursă dezvoltat de Microsoft, lansat oficial în anul 2015. Este gratuit, open-source și cross-platform, ceea ce înseamnă că poate fi utilizat pe Windows, macOS și Linux. VS Code s-a impus rapid ca unul dintre cele mai populare editoare de cod din lume datorită simplității sale, performanței excelente și ecosistemului extins de extensii.

Editorul oferă suport încorporat pentru un număr mare de limbaje de programare, cum ar fi JavaScript, Python, C++, Java, HTML/CSS și altele. În plus, prin extensii din marketplace, utilizatorii pot adăuga suport pentru tehnologii și limbaje suplimentare, pot integra debuggere, terminale, controlul versiunilor (Git) sau chiar medii de dezvoltare virtuale.

Visual Studio Code este folosit frecvent în mediul universitar, mai ales în cadrul facultăților cu profil informatic. Studenții îl utilizează pentru scrierea și testarea codului în diverse limbaje de programare, fiind preferat pentru interfața intuitivă, instalarea rapidă și compatibilitatea cu platforme educaționale. De asemenea, multe cursuri și laboratoare recomandă sau solicită folosirea acestui editor, datorită integrării sale eficiente cu GitHub, terminalul integrat și suportul pentru debugging, facilitând astfel procesul de învățare și dezvoltare.

2.2 Java

Java este un limbaj de programare de nivel înalt, dezvoltat de compania Sun Microsystems la mijlocul anilor 1990, sub coordonarea lui James Gosling, și lansat oficial în anul 1995. Limbajul a fost creat cu scopul de a oferi portabilitate, siguranță și simplitate, principiul său de bază fiind „write once, run anywhere”, ceea ce permite rularea aplicațiilor Java pe diferite platforme fără modificări majore ale codului. De-a lungul timpului, Java a evoluat constant, devenind unul dintre cele mai utilizate limbaje de programare din lume, folosit în dezvoltarea de aplicații desktop, web, mobile și jocuri.

Java este un limbaj orientat pe obiect, conceput pentru a încuraja organizarea clară a codului și reutilizarea acestuia prin intermediul claselor, obiectelor, moștenirii și polimorfismului. Spre deosebire de limbajele care oferă acces direct la memorie, Java gestionează automat memoria prin intermediul mecanismului de garbage collection, reducând riscul erorilor precum scurgerile de memorie. Această abordare face ca Java să fie mai sigur și mai ușor de utilizat, chiar dacă presupune un control mai redus asupra resurselor hardware comparativ cu limbaje precum C++.

În mediul universitar, Java este frecvent utilizat ca limbaj de bază pentru predarea programării orientate pe obiect. Datorită sintaxei clare și a ecosistemului bogat de biblioteci standard, Java este considerat potrivit pentru înțelegerea conceptelor fundamentale ale programării, precum structuri de control, tipuri de date, clase, interfețe și gestionarea evenimentelor. Prin intermediul proiectelor practice, studenții pot dezvolta aplicații complexe și interactive, dobândind atât cunoștințe teoretice, cât și experiență practică relevantă pentru domeniul dezvoltării software.

3. Scenariu de functionare

3.1 Scurtă prezentare

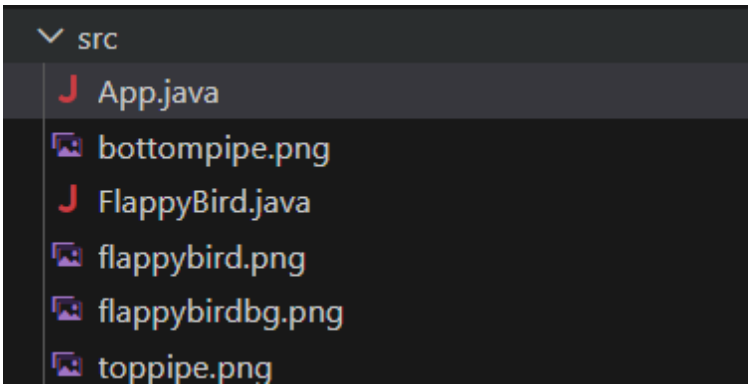
Acest program reprezintă o implementare simplă, dar funcțională, a jocului Flappy Bird, dezvoltată în limbajul Java, folosind biblioteca Swing pentru interfața grafică.

Jocul se desfășoară într-o fereastră de tip grafic, în care jucătorul controlează o pasăre ce trebuie să evite obstacole reprezentate de țevile verticale care apar la intervale regulate. Scopul este să treci cât mai departe posibil, acumulând puncte pentru fiecare set de țevile depășit fără coliziuni.

Pasărea se mișcă constant spre dreapta, iar jucătorul poate apăsa tasta SPACE pentru a sări, controlând astfel altitudinea acesteia. Obstacolele sunt generate aleatoriu, iar distanța dintre țevile superioare și inferioare formează culoare prin care pasărea trebuie să treacă. Jocul se termină fie când pasărea lovește o țevă sau cade în afara ecranului, fie când utilizatorul decide să închidă aplicația. După terminarea jocului, jucătorul poate apăsa SPACE pentru a reseta și a începe o nouă rundă.

Programul se folosește de concepte fundamentale din programare, precum clase și obiecte, liste și colecții, generare aleatorie, gestionarea evenimentelor (input de la tastatură) și actualizarea periodică a stării jocului prin timere, oferind un exemplu complet de joc 2D interactiv implementat în Java.

3.2 Prezentare detaliată



Programul este alcătuit din două clase principale, App și FlappyBird, care colaborează pentru a crea și rula jocul Flappy Bird într-o interfață grafică. Structura aplicației este simplă

și clar delimitată: clasa App se ocupă de inițializarea ferestrei, iar clasa FlappyBird gestionează logica jocului, desenarea elementelor și interacțiunea cu utilizatorul.

3.2.1. Inițializarea aplicației și a ferestrei grafice (App.java)

```
public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        int boardWidth = 360;
        int boardHeight = 640;

        JFrame frame = new JFrame(title: "Flappy Bird");
        // frame.setVisible(true);
        frame.setSize(boardWidth, boardHeight);
        frame.setLocationRelativeTo(c: null);
        frame.setResizable(resizable: false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Acest fragment reprezintă punctul de pornire al aplicației. Sunt definite dimensiunile ferestrei de joc și este creat un obiect de tip JFrame, care va conține întreaga interfață grafică. Fereastra este centrată pe ecran, are dimensiune fixă și este configurată astfel încât aplicația să se închidă complet atunci când utilizatorul închide fereastra.

```
        FlappyBird flappyBird = new FlappyBird();
        frame.add(flappyBird);
        frame.pack();
        flappyBird.requestFocus();
        frame.setVisible(b: true);
    }
}
```

În această parte este creată instanța clasei FlappyBird, care conține logica și desenarea jocului. Panoul este adăugat în fereastră, iar metoda requestFocus() permite captarea inputului de la tastatură. Prin apelul setVisible(true), fereastra devine vizibilă și jocul pornește efectiv.

3.2.2. Definirea clasei principale de joc (FlappyBird)

```
public class FlappyBird extends JPanel implements ActionListener, KeyListener {
    int boardWidth = 360;
    int boardHeight = 640;
```

Clasa FlappyBird extinde JPanel, ceea ce permite desenarea elementelor grafice în fereastră. Implementarea interfețelor ActionListener și KeyListener oferă posibilitatea de a gestiona atât bucla principală de joc, cât și inputul de la tastatură.

3.2.3. Resurse grafice și poziții inițiale

```
//images
Image backgroundImg;
Image birdImg;
Image topPipeImg;
Image bottomPipeImg;

//bird class
int birdX = boardWidth/8;
int birdY = boardHeight/2;
int birdWidth = 34;
int birdHeight = 24;
```

Aici sunt declarate imaginile utilizate în joc, precum și poziția și dimensiunea inițială a păsării. Pasărea este plasată în partea stângă a ecranului, la o înălțime medie, pentru a oferi jucătorului timp de reacție la începutul jocului.

3.2.4. Reprezentarea obiectelor de joc: pasărea și țevile

```
class Bird {
    int x = birdX;
    int y = birdY;
    int width = birdWidth;
    int height = birdHeight;
    Image img;

    Bird(Image img) {
        this.img = img;
    }
}
```

Clasa internă Bird modelează pasărea controlată de jucător. Aceasta reține poziția, dimensiunea și imaginea, fiind utilizată atât în logica jocului, cât și la desenare.

```
//pipe class
int pipeX = boardWidth;
int pipeY = 0;
int pipeWidth = 64;  //scaled by 1/6
int pipeHeight = 512;

class Pipe {
    int x = pipeX;
    int y = pipeY;
    int width = pipeWidth;
    int height = pipeHeight;
    Image img;
    boolean passed = false;

    Pipe(Image img) {
        this.img = img;
    }
}
```

Clasa Pipe reprezintă obstacolele jocului. Variabila passed indică dacă pasărea a trecut deja de obstacol și este folosită pentru calcularea scorului.

3.2.5. Variabile de control și stare a jocului

```
//game logic
Bird bird;
int velocityX = -4; //move pipes to the left speed (simulates bird moving right)
int velocityY = 0; //move bird up/down speed.
int gravity = 1;

ArrayList<Pipe> pipes;
Random random = new Random();

Timer gameLoop;
Timer placePipeTimer;
boolean gameOver = false;
double score = 0;
```

Aceste variabile controlează mișcarea păsării și a obstacolelor, starea jocului și scorul. Gravitația determină coborârea păsării, iar deplasarea țevilor spre stânga simulează înaintarea în joc.

3.2.6. Constructorul clasei și inițializarea jocului

```
FlappyBird() {
    setPreferredSize(new Dimension(boardWidth, boardHeight));
    // setBackground(Color.blue);
    setFocusable(focusable: true);
    addKeyListener(this);
```

Constructorul configurează panoul de joc, permițând captarea inputului de la tastatură.

```
//load images
backgroundImg = new ImageIcon(getClass().getResource(name: "./flappybirdbg.png")).getImage();
birdImg = new ImageIcon(getClass().getResource(name: "./flappybird.png")).getImage();
topPipeImg = new ImageIcon(getClass().getResource(name: "./toppipe.png")).getImage();
bottomPipeImg = new ImageIcon(getClass().getResource(name: "./bottompipe.png")).getImage();
```

Imaginile jocului sunt încărcate din resursele proiectului.

```
//bird
bird = new Bird(birdImg);
pipes = new ArrayList<Pipe>();
```

Este creată pasărea și lista care va conține obstacolele.

```
//place pipes timer
placePipeTimer = new Timer(delay: 1500, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      // Code to be executed
      placePipes();
    }
});
placePipeTimer.start();

//game timer
gameLoop = new Timer(1000/60, this); //how long it takes to st
gameLoop.start();
```

Sunt inițializate cele două timere: unul pentru generarea obstacolelor și unul pentru bucla principală de joc.

3.2.7. Generarea obstacolelor

```
void placePipes() {
    //(0-1) * pipeHeight/2.
    // 0 -> -128 (pipeHeight/4)
    // 1 -> -128 - 256 (pipeHeight/4 - pipeHeight/2) = -3/4 pipeHeight
    int randomPipeY = (int) (pipeY - pipeHeight/4 - Math.random()*(pipeHeight/2));
    int openingSpace = boardHeight/4;

    Pipe topPipe = new Pipe(topPipeImg);
    topPipe.y = randomPipeY;
    pipes.add(topPipe);

    Pipe bottomPipe = new Pipe(bottomPipeImg);
    bottomPipe.y = topPipe.y  + pipeHeight + openingSpace;
    pipes.add(bottomPipe);
}
```

Această metodă generează un set de două țevi poziționate aleatoriu pe axa verticală, păstrând un spațiu constant între ele.

3.2.8. Desenarea elementelor grafice

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    draw(g);
}
```

```
public void draw(Graphics g) {
    //background
    g.drawImage(backgroundImg, x: 0, y: 0, this.boardWidth, this.boardHeight, observer: null);

    //bird
    g.drawImage(birdImg, bird.x, bird.y, bird.width, bird.height, observer: null);

    //pipes
    for (int i = 0; i < pipes.size(); i++) {
        Pipe pipe = pipes.get(i);
        g.drawImage(pipe.img, pipe.x, pipe.y, pipe.width, pipe.height, observer: null);
    }
```

Această parte se ocupă de afișarea fundalului, păsării și obstacolelor.

```
    //score
    g.setColor(Color.white);

    g.setFont(new Font(name: "Arial", Font.PLAIN, size: 32));
    if (gameOver) {
        g.drawString("Game Over: " + String.valueOf((int) score), x: 10, y: 35);
    }
    else {
        g.drawString(String.valueOf((int) score), x: 10, y: 35);
    }

}
```

Este afișat scorul curent sau mesajul de final.

3.2.9. Mișcarea și logica jocului

```
public void move() {
    //bird
    velocityY += gravity;
    bird.y += velocityY;
    bird.y = Math.max(bird.y, b: 0);
```

Aplică gravitația și limitează poziția păsării în partea superioară a ecranului.

```
    //pipes
    for (int i = 0; i < pipes.size(); i++) {
        Pipe pipe = pipes.get(i);
        pipe.x += velocityX;

        if (!pipe.passed && bird.x > pipe.x + pipe.width) {
            score += 0.5; //0.5 because there are 2 pipes!
            pipe.passed = true;
        }

        if (collision(bird, pipe)) {
            gameOver = true;
        }
    }

    if (bird.y > boardHeight) {
        gameOver = true;
    }
}
```

Deplasează obstacolele, actualizează scorul și verifică condițiile de finalizare a jocului.

3.2.10. Detectarea coliziunilor

```
boolean collision(Bird a, Pipe b) {
    return a.x < b.x + b.width &&
           a.x + a.width > b.x &&
           a.y < b.y + b.height &&
           a.y + a.height > b.y;
}
```

Metoda verifică suprapunerea dreptunghiurilor asociate păsării și obstacolelor. Dacă există o intersecție, jocul este încheiat.

3.2.11. Bucla principală și inputul utilizatorului

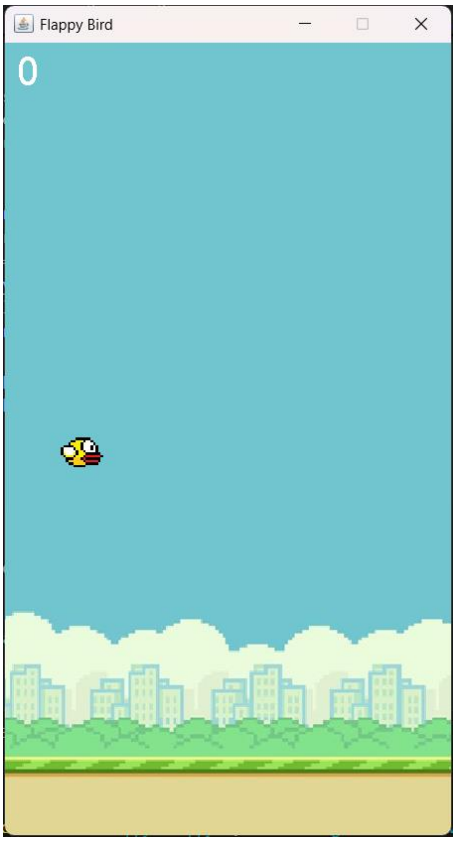
```
public void actionPerformed(ActionEvent e) {
    move();
    repaint();
    if (gameOver) {
        placePipeTimer.stop();
        gameLoop.stop();
    }
}
```

Această metodă este apelată periodic de bucla principală de joc și actualizează starea aplicației.

```
public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_SPACE) {
        // System.out.println("JUMP!");
        velocityY = -9;

        if (gameOver) {
            //restart game by resetting conditions
            bird.y = birdY;
            velocityY = 0;
            pipes.clear();
            gameOver = false;
            score = 0;
            gameLoop.start();
            placePipeTimer.start();
        }
    }
}

//not needed
@Override
public void keyTyped(KeyEvent e) {}

@Override
public void keyReleased(KeyEvent e) {}
```

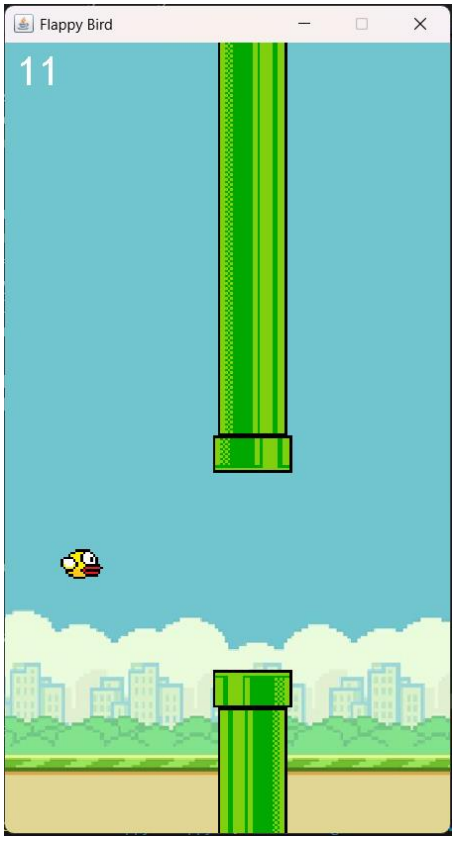
Gestionează controlul păsării și permite reluarea jocului după finalizare.

3.3. Ilustrarea funcționării aplicației



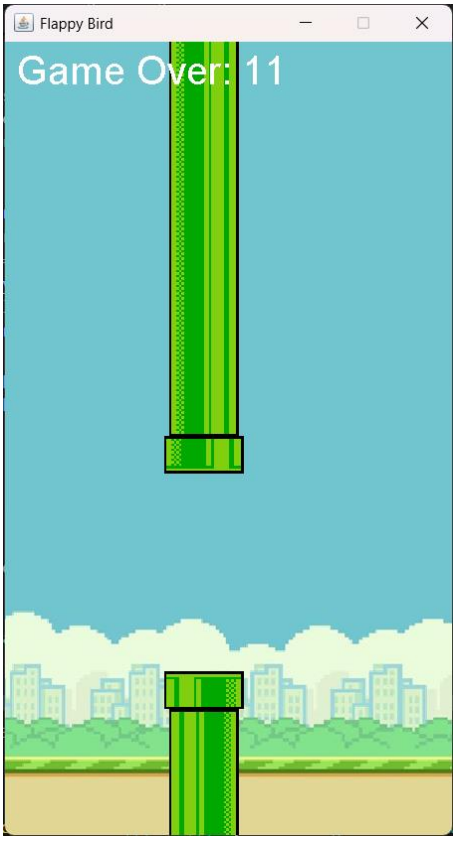
Inițializarea jocului

Această imagine prezintă starea inițială a jocului Flappy Bird, imediat după lansarea aplicației. Pasărea este poziționată în partea stângă a ecranului, iar fundalul și zona de joc sunt încărcate corespunzător. În acest moment, jocul este pregătit pentru a primi input de la utilizator.



Jocul în desfășurare

În această captură este ilustrată desfășurarea normală a jocului, cu pasărea în mișcare și obstacolele generate aleatoriu. Scorul este afișat în partea superioară a ecranului și este actualizat în timp real pe măsură ce pasărea trece de obstacole. Controlul păsării se realizează prin apăsarea tastei SPACE.



Detectarea coliziunii și finalizarea jocului

Această imagine surprinde momentul în care pasărea intră în coliziune cu un obstacol sau părăsește zona de joc. Aplicația afișează mesajul „Game Over” și scorul obținut, iar bucla principală de joc este oprită. Utilizatorul poate relua jocul prin apăsarea tastei SPACE.

4. Concluzie

Proiectul dezvoltat (implementarea jocului Flappy Bird în limbajul Java) a reprezentat un exercițiu complet de aplicare a conceptelor fundamentale de programare orientată pe obiect și de dezvoltare a aplicațiilor grafice. Scopul principal al proiectului a fost recrearea mecanicilor esențiale ale jocului clasic într-un mod funcțional, stabil și intuitiv, oferind o experiență de joc fluidă și corectă, respectând totodată principiile de organizare clară a codului și separare a responsabilităților între componentele aplicației.

Pe parcursul dezvoltării aplicației, au fost utilizate concepte fundamentale precum programarea orientată pe obiect (prin definirea claselor pentru pasăre și obstacole), utilizarea colecțiilor pentru gestionarea elementelor dinamice ale jocului, generarea aleatorie a obstacolelor, detectarea coliziunilor și gestionarea timpului prin intermediul temporizatoarelor. De asemenea, a fost implementată tratarea inputului de la tastatură și actualizarea periodică a stării jocului, asigurând o desfășurare coerentă și predictibilă a rundelor de joc.

Funcționalitățile cheie ale aplicației includ:

-controlul pasării prin intermediul tastaturii și simularea gravitației;

-generarea aleatorie a obstacolelor și deplasarea acestora pe ecran;

-detectarea coliziunilor dintre pasăre și obstacole sau marginile ecranului;

-afișarea în timp real a scorului și a stării jocului;

-posibilitatea de reluare a jocului după terminarea unei runde, fără repornirea aplicației.

Din punct de vedere educațional, realizarea acestui proiect a oferit o oportunitate valoroasă pentru consolidarea cunoștințelor de programare în Java, precum și pentru înțelegerea modului în care conceptele teoretice pot fi aplicate într-un context practic și interactiv. Accentul a fost pus pe claritatea codului, organizarea logică a claselor și utilizarea eficientă a mecanismelor oferite de biblioteca Swing.

În concluzie, aplicația demonstrează capacitatea de a implementa un joc 2D complet funcțional într-un mediu grafic, utilizând concepte de bază din programarea Java. Jocul Flappy Bird realizat poate fi extins cu ușurință în viitor, prin adăugarea unor niveluri de dificultate variabile, îmbunătățiri grafice, efecte sonore sau un sistem de salvare a scorurilor, constituind astfel o bază solidă pentru proiecte viitoare de complexitate mai ridicată.

5. Bibliografie

https://docs.oracle.com/en/java/javase/

https://docs.oracle.com/javase/tutorial/uiswing/

https://en.wikipedia.org/wiki/Game_loop

https://zetcode.com/javagames/

https://www.youtube.com/watch?v=Xw2MEG-FBsE