



UNIVERSITATEA TEHNICĂ GHEORGHE ASACHI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Vânătoarea de diamante - JOC 2D -

STUDENTI

Paraschiv Florin-Vlăduț, 1409B
Prelipcean Dragoș-Iulian, 1409B

Contents

1 Descriere proiect	3
2 Tehnologii și dependențe utilizate	4
2.1 Tehnologii	4
2.2 Dependente	4
2.2.1 Hibernate și PostgreSQL	4
2.2.2 Project Lombok	4
3 Interacțiunea cu utilizatorul	5
4 Descrierea tehnică a proiectului	5
4.1 Mod de funcționare	5
4.1.1 Actualizarea ferestrei	5
4.1.2 Randarea animațiilor	5
4.1.3 Trecerea la următorul nivel	5
4.1.4 Coliziunea dintre obiectele grafice	6
4.1.5 Construcția hărților	7
4.2 Portabilitate	8
4.2.1 Ce este docker?	8
4.2.2 Ce este un container?	8
4.2.3 Cum ne folosim de docker în aplicație	8
5 Mod instalare joc	8
5.1 Instalare Docker	8
5.2 Instalare Java 11	9
5.3 Descărcarea jocului	9
5.4 Pornire joc	9
6 Componența echipei	9
7 Codul sursa	10
8 Bibliografie	10

1 Descriere proiect

Proiectul „Vânătoarea de diamante” constă într-un joc 2D, top-down, oferă o vedere de sus, de tip single-player. La pornirea jocului, utilizatorului îi este prezentat un meniu principal din care poate să aleagă una din cele 5 opțiuni, și anume Start, Settings, Score, About și Exit după cum se poate observa în figurile de mai jos.



Figure 1: Meniul principal

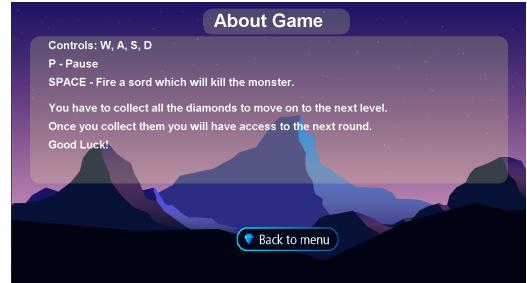


Figure 2: Despre joc

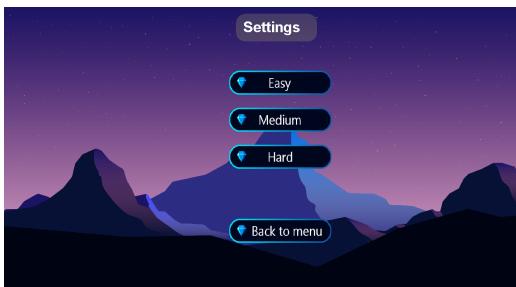


Figura 3: Setari



Figura 4: Scoruri

La apăsarea butonului de start se va deschide o fereastră în care utilizatorul își va introduce numele, urmând să înceapă jocul.

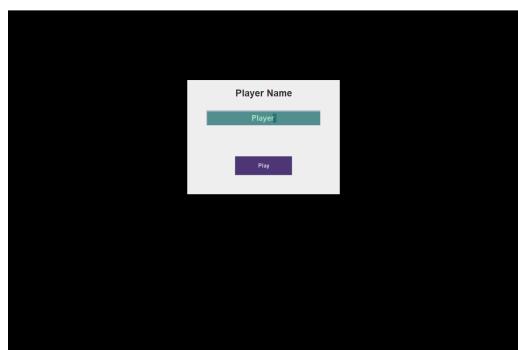


Figura 5: Alegerea numelui

La început, primim un caracter care are rolul de a colecta diamante de pe mapă în cel mai scurt timp posibil. De-a lungul jocului, jucătorul trebuie să se ferească de monștri și, ca mod de apărare, poate arunca cu sulițe în ei pentru a-i doborâ. Jucătorul este obligat să colecteze toate diamantele de pe mapă pentru a trece la urmatorul nivel. Jocul este construit pe două nivele, caracterul având trei viață la începutul jocului. Jocul se termină atunci când jucătorul trece prin poarta de la nivelul doi, cu toate diamantele colectate.

2 Tehnologii și dependențe utilizate

2.1 Tehnologii

- IntelliJ: s-a folosit drept IDE;
- Java JDK 11: versiunea de java a proiectului;
- Maven: s-a folosit pentru a crea JAR-ul cu dependențe;
- Docker: pentru portabilitate;
- PostgreSQL 14: baza de date în care stocăm scorul jucătorului;
- PageAdmin 4: tool pentru a putea vizualiza și modela datele din baza de date;
- Git: pentru versionarea proiectului și o mai bună colaborare în echipă.

2.2 Dependențe

2.2.1 Hibernate și PostgreSQL

Hibernate este un framework care simplifică dezvoltarea aplicațiilor Java care comunică cu o bază de date.

În figura de mai jos se pot observa proprietățile folosite în proiect pentru a ne putea conecta la o bază de date Postgres.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<.hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQL82Dialect</property>
        <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
        <property name="hibernate.connection.url">jdbc:postgresql://localhost:5433/piu_project</property>
        <property name="hibernate.connection.username">pq_piu</property>
        <property name="hibernate.connection.password">pq_piu</property>

        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <property name="hbm2ddl.auto">update</property>

        <mapping class="PIUGame.Database.Table.Record"/>
    </session-factory>
</hibernate-configuration>
```

Figura 6: Structura fișierului hibernate.cfg.xml

2.2.2 Project Lombok

Project Lombok este o librărie care este folosită pentru a minimiza/elimina secțiuni de cod care sunt repetate în mai multe locuri, cu puține sau deloc variații, și pentru a salva timp dezvoltatorilor în timpul dezvoltării prin folosirea adnotărilor.

3 Interacțiunea cu utilizatorul

Utilizatorul va putea vizualiza numărul de diamante colectate și numărul de vieți rămase. De asemenea, va avea opțiunea de a pune pauză jocului, fiind salvată starea în care se află jucătorul la momentul respectiv. Inamicii vor fi mereu pe urmele lui, iar la contactul cu eroul acesta își va pierde o viață. Trecerea la următorul nivel va fi posibilă doar dacă toate diamantele, din acel nivel, vor fi colectate de utilizator.

Comenzile prin care utilizatorul va interacționa cu caracterul său sunt tastele W, A, S, D pentru deplasare și tasta Space, pentru acțiunea de atac asupra inamicilor.

4 Descrierea tehnică a proiectului

4.1 Mod de funcționare

4.1.1 Actualizarea ferestrei

Jocul este construit pe State-uri, cele mai importante dintre acestea fiind: MenuState, PlayState, SettingsState, ScoreState, ResumeGameState, LoseState, FinishedGameState. Prima stare care ne este prezentată la deschiderea acestuia este starea de meniu, urmând ca utilizatorul să aleagă ce dorește să facă mai departe.

Fiecare din aceste State-uri conține mai multe obiecte care se desenează pe un Canvas. Acest canvas este creat la pornirea programului și se actualizează cu o rată de 60 de ori pe secundă(60fps), adică fiecare frame durează 16.6ms. La fiecare actualizare se șterge tot conținutul din fereastră și se redesenează cu noile informații.

4.1.2 Randarea animațiilor

În cadrul unei hărți există mai multe tipuri de obiecte, cele care se desenează de fiecare dată la fel, spre exemplu un obiect de tip copac și cele care se compun din mai multe cadre, iar când sunt redate foarte repede crează efectul unei animații. Modul acesta de afișare este folosit pentru desenarea player-ului, monștrilor și exploziilor.

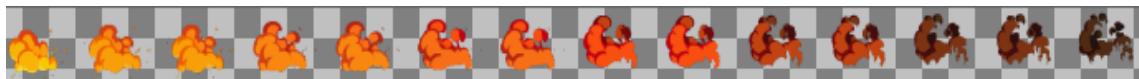


Figura 7: Animatie explozie

Acesta funcționează în urmatorul fel. Pentru explicație vom folosi animația de tip explozie. Imaginea încarcată conține mai multe cadre ale efectului după cum se poate observa în figură. Fiecare cadru se taie din imagine și se salvează într-un vector de frame-uri. În funcție de viteza cu care dorim să fie afișat efectul, vom actualiza care din aceste cadre se desenează, iar când animația se termină există posibilitatea de a fi redată din nou, cum se întamplă pentru caractere, sau pur și simplu se termină, cum este în cazul exploziilor, redându-se astfel un efect cursiv.

4.1.3 Trecerea la următorul nivel

Pentru trecerea la următorul nivel jucătorul trebuie să colecteze toate diamantele existente pe hartă, altfel nu îi este permis să meargă mai departe. Această acțiune se realizează printr-o serie de verificări asupra coordonatelor cum se poate observa în figura de mai jos.

Pentru cazul în care jucătorul a colectat toate diamantele și trece de linia neagră atunci caracterul este "tras" automat către zona marcată cu roșu, iar apoi se desenează următorul nivel.

În cazul în care jucătorul nu a colectat toate diamantele și trece de linia neagră, caracterul nu este dus către centrul porții și își poate continua deplasarea pe hartă, însă pentru a se dezactiva starea ce marchează trecerea către nivelul urmator, player-ul trebuie să iasă din zona marcată cu linie albă. Aceste verificări nu influențează cu nimic parcursul jucătorului.

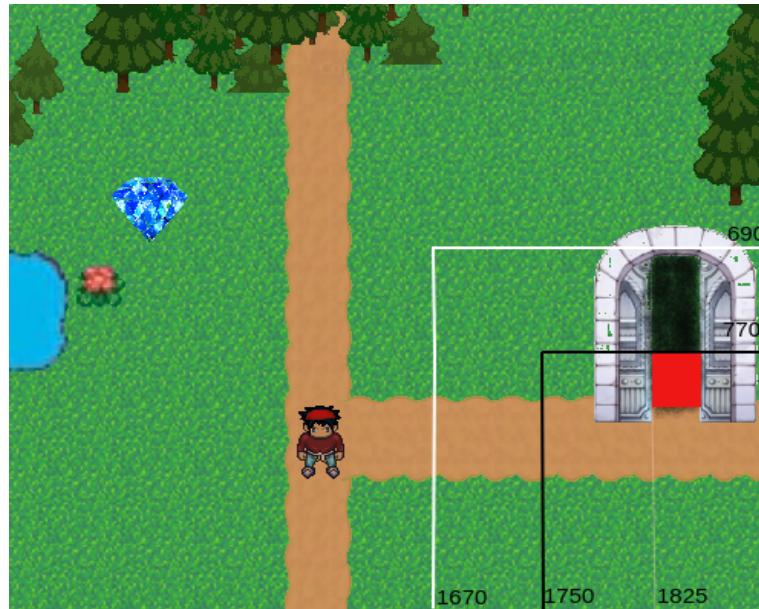


Figura 8: Trecerea la nivelul urmator

4.1.4 Coliziunea dintre obiectele grafice

Pentru coliziunile dintre obiecte se folosesc dreptunghiuri ce mărginesc imaginea. Dacă două dreptunghiuri se suprapun parțial la un moment dat atunci se consideră că ele intră în coliziune. Când jucătorul se intersecează cu unul din monștrii atunci acesta își pierde o viață și este readus în poziția de început a jocului. Când un diamant intră în coliziune cu caracterul eroului, diamantul este colectat și dispără de pe hartă. În prima imagine de mai jos se poate observa evidențierea dreptunghiurilor ce sunt folosite la calculul coliziunilor pentru diamant cât și pentru caracter, iar în dreapta modul în care se realizează suprapunerea.



Figura 9: Coliziunea caracterului cu un diamant

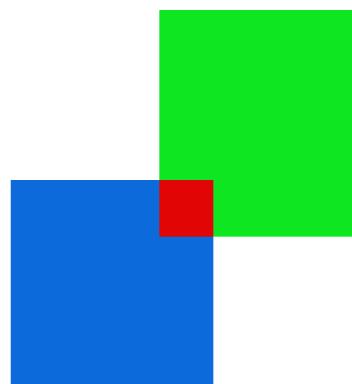


Figura 10: Coliziune dintre două dreptunghiuri

4.1.5 Construcția hărților

Pentru deplasarea caracterelor pe hartă s-a folosit o matrice alcătuită din valori de 0 și 1, în care 0 reprezintă locul prin care este permisă trecerea, iar 1 pe unde nu este permisă. Fiecare element din matrice reprezintă o dala cu dimensiunea de 48x48 de pixeli. Pe parcursul jocului se va lua mereu în calcul poziția caracterelor care se deplasează pe hartă și în urma unor calcule se va stabili dacă obiectele se află pe o pozitie de 0 sau au întâlnit o dala de tip solid, reprezentată de id-ul 1, ceea ce nu le permite trecerea. De remarcat este faptul că imaginea ce reprezintă o dala, adică un element din matrice, este aceeași atât pentru 0 cât și pentru 1, ceea ce înseamnă că existădale de tip iarbă peste care se poate trece și dale de tip iarbă peste care nu este permisă trecerea.

Pentru harta de la primul nivel s-au folosit dale de tip iarbă, iar peste acestea s-au adăugat, cu ajutorul coordonatelor, imagini ce reprezintă diferite obiecte, precum, copaci, case sau drumuri. O exemplificare mai clară se poate observa în figura de mai jos.



Figura 11: Harta nivel 1

Harta de la nivelul al doilea a fost construită în Photoshop în modul următor: s-a tinut cont de faptul că o dala este de 48x48 și s-a creat o imagine în care fiecare patrat este de 48x48 pentru ca player-ul să aibă o coliziune cat mai bună cu dalele de tip solid. Astfel, toata harta este o imagine suprapusă peste matricea alcătuită din valori de 0 și 1.

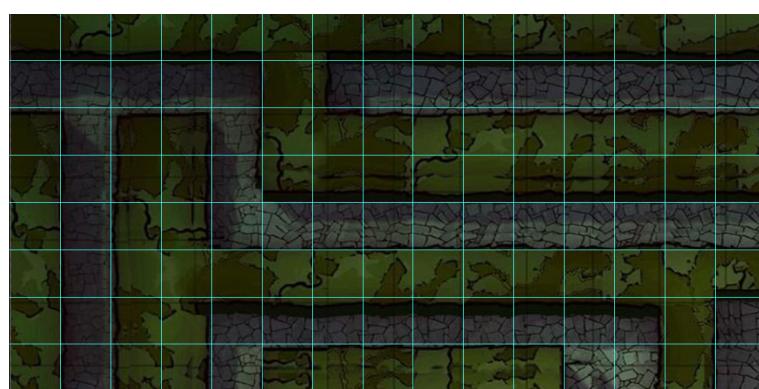


Figura 12: Harta nivel 2

4.2 Portabilitate

4.2.1 Ce este docker?

Docker-ul este o platformă de tipul open-source folosită pentru construirea, lansarea și administrarea containerelor. Permite dezvoltatorilor să împacheteze aplicații în containere, componente executabile standardizate care combină codul sursă al aplicației cu bibliotecile sistemului de operare și dependențele necesare pentru a rula acel cod în orice mediu.

4.2.2 Ce este un container?

Un container este o unitate standard de software care împachetează codul și toate dependențele acestuia, astfel încât aplicația să ruleze la schimbarea de la un mediu de calcul la altul. Disponibil atât pentru aplicațiile Linux, cât și Windows, containerul va rula întotdeauna la fel, indiferent de infrastructură, izolând software-ul de mediul său și asigurând funcționarea uniformă pe orice sistem de operare.

4.2.3 Cum ne folosim de docker în aplicație

In interiorului proiectului este un fișier docker-compose.yml care are următoarea structură:

```
version: "3.7"

services:
  postgresdb:
    image: postgres:14
    container_name: database
    restart: always
    command:
      -p 5433
    environment:
      POSTGRES_DB: piu_project
      POSTGRES_USER: pq_piu
      POSTGRES_PASSWORD: pq_piu
    ports:
      - 5433:5433
```

Figura 13: Structura fisierului docker-compose.yml

Portabilitatea este obținută prin crearea unui container pornind de la o imagine de bază de date, i.e. noi ne folosim de o imagine Postgres, versiunea 14. În continuare, am adăugat variabile de environment pentru inițializarea unei baze de date cu numele *piu_project* și credențialele pentru **superuser**. Pe lângă credențiale, s-au mai adăugat și câteva setări suplimentare, anume portul pe care ruleaza containerul și portul pe care ruleaza baza de date din interiorul containerului.

5 Mod instalare joc

5.1 Instalare Docker

Pentru instalarea Docker-ului se acceseaza [acest link](#) și se instaleaza versiunea conform sistemului de operare.

5.2 Instalare Java 11

Pentru instalarea Java 11 se acceseaza [acest link](#).

5.3 Descărcarea jocului

Se acceseaza [acest link](#) si se va descărca tot fisierul, care include: jar-ul, docker-compose.yml si executabilul.

5.4 Pornire joc

- Se verifică dacă docker-ul este pornit
- Se pornește jocul prin rularea executabilului descărcat la pasul anterior

Dacă pașii au fost urmați, se va deschide un CMD urmat de interfata jocului, ca în figurile de mai jos:

```
D:\Programs\repository\Project_PDU_Version_1\playgame$ docker compose up -d
Starting database ... done
D:\Programs\repository\Project_PDU_Version_1\playgame$ ./jar VanuatuReindeerGame.jar
Dec 12, 2021 7:13:18 PM org.hibernate.Version logVersion
INFO: HHH000412: Recognized obsolete hibernate namespace http://hibernate.sourceforge.net/hibernate-configuration. Use namespace http://www.hibernate.org/
Dec 12, 2021 7:13:18 PM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Core [hibernate.cfg.xml] configuration loaded
Dec 12, 2021 7:13:19 PM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Configuration resource [hibernate.cfg.xml] loaded
Dec 12, 2021 7:13:19 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager cList
INFO: HHH000039: Relaxed restrictions on collection types
Dec 12, 2021 7:13:20 PM org.hibernate.connection.ConnectionManagerImpl configure
INFO: HHH000406: Using alternate built-in connection pool (not for production use!)
Dec 12, 2021 7:13:20 PM org.hibernate.connection.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH000408: using driver [org.postgresql.Driver] at [jdbc:postgresql://localhost:5432/pdu_project]
Dec 12, 2021 7:13:20 PM org.hibernate.connection.DriverManagerConnectionProviderImpl buildCreate
INFO: HHH000408: Connection properties: [password='***', user='postgres']
Dec 12, 2021 7:13:20 PM org.hibernate.connection.DriverManagerConnectionProviderImpl buildDriver
INFO: HHH000408: Acquired native JDBC connection
Dec 12, 2021 7:13:20 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl<init>
INFO: HHH000408: Acquired native JDBC connection
Dec 12, 2021 7:13:20 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000408: Using dialect [org.hibernate.dialect.PostgreSQLDialect]
Dec 12, 2021 7:13:21 PM org.hibernate.resource.transaction.backend.jdbc.internal.JdbcTransactionIsolationManager$IsolateConnection
INFO: HHH000410: Connection obtained from JdbcConnectionPool [org.hibernate.engine.jdbc.internal.JdbcConnectionPool@1f3e1f1] will be set into auto-commit mode
Dec 12, 2021 7:13:21 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initializeService
INFO: HHH000408: Using JtaPlatform implementation [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
game created.
```

Figura 14: CMD

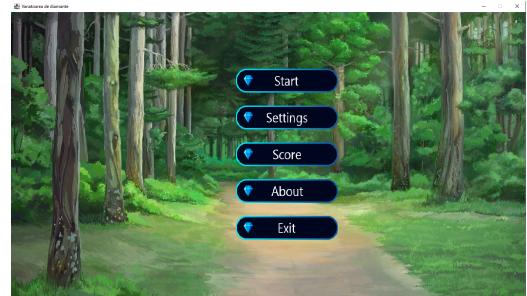


Figura 15: Meniul principal

6 Componența echipei

Echipa este formată din două persoane, fiecare cu următoarele atribuții:

- Prelipcean Dragoș-Iulian:
 - controlul jucătorului și al inamicilor;
 - modul de urmărire a jucătorului de către inamici;
 - gestiunea nivelurilor și traseea jucătorului de la un nivel la altul;
 - modul de pauză al jocului și numărul de vieți rămase;
 - editarea elementelor din care va fi constituită interfața(harta) pentru nivelul 1;
 - gestionarea elementelor legată de meniurile jocului
- Paraschiv Florin-Vlăduț:
 - crearea legăturii cu baza de date;
 - editarea elementelor din care va fi constituită interfața(harta) pentru nivelul 2;
 - logica pentru partea de scor, timer și salvare a progresului jucătorului;
 - crearea container-ului în care este salvată baza de date
 - stabilirea dependențelor necesare funcționării jocului și modul de instalare a acestuia

7 Codul sursă

Codul sursă al proiectului se poate vizualiza accesând [acest link](#)

8 Bibliografie

Docker: <https://www.docker.com/>

Maven: <https://maven.apache.org/guides/>Hibernate: <https://hibernate.org/orm/documentation/5.6/>PostgreSQL: <https://www.postgresql.org/docs/14/index.html>Pgadmin 4: <https://www.pgadmin.org/docs/pgadmin4/development/index.html>Project Lombok: <https://projectlombok.org/features/all>Git: <https://git-scm.com/doc>Java windows: <https://www.javatpoint.com/java-jframe>Elemente grafice: https://www.tutorialspoint.com/awt/awt_graphics_class.htm