UNIVERSITATEA “ȘTEFAN CEL MARE” SUCEAVA
FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR
SPECIALIZAREA CALCULATOARE

PROIECT DISCIPLINA POO

Joc video – Dino Run

Autor: Vărăreanu Vasile-Olivian, grupa 3122B

TEMA PROIECT

Tema proiectului este crearea unui joc de tip side-scrolling în care personajul principal este un dinozaur ce pornește într-o călătorie iar pe drum întâlnește obstacole precum cactuși.. Am fost inspirat să aleg acest joc mulțumită browserului Google Chrome, care afișează acest joc în momentul în care există o problemă la accesarea internetului.

Jocul este implementat în limbajul de programare JavaScript însă eu am încercat o recreare cât mai fidelă a acestui joc în limbajul C++, folosind librăria SFML, ce cuprinde facilități în crearea designurilor și interacțiunilor grafice, respectiv audio.

CUPRINS

1. ABORDAREA TEORETICA

În primul rând această aplicație va rula în interiorul unei ferestre cu rezoluția 1000 x 500 pixeli. Pentru a crea lumea jocului vom avea nevoie de caracterul principal, obstacole precum cactuși, care vor fi amplasați în mod aleatoriu pe pământ iar pentru partea de design și ambianță voi adăuga nori.

Singurul control de care are nevoie caracterul principal este pentru salt, control ce va fi operat de către tasta “space”. În momentul în care se va produce o coliziune cu unul dintre obstacole, jocul se va termina, salvându-se scorul obținut și totodată, se poate face o comparație între scoruri. Scopul jocului este de a obține un scor cât mai mare.

Pentru o distribuire a jocului va fi necesară crarea unui fișier executabil, alături de resursele necesare precum imaginile pentru texturi, sunete, headere și librării pentru a putea fi rulat atât pe Windows, cât și pe Linux.

2. ELEMENTE SPECIFICE POO

Codul sursă va fi împărțit în mai multe clase diferite, fiecare având rolul său specific. Vom avea următoarele clase: Ground (pământul), Obstacle (obstacol), Obstacles (ce va fi totodată și o clasă ce conține o listă de clase Obstacle), Dino (caracterul principal) , Scores (ce va reține scorurile), Clouds (nori), GameState (statutul jocului în care se instanțează fiecare clasă), clasa RestartButton (pentru reînceperea jocului) și clasa FPS care va fi folosită pentru afișarea numărului de cadre pe secundă pentru o vizualizare a nivelului de optimizare. De asemenea am

folosit conceptul de ierarhizare a metodelor specifice clasei sf. precum : sf::Event, sf::Time, sf::Clock.

3. IMPLEMENTARE

Pentru implementarea acestei aplicații vom avea nevoie mai întâi să instalăm librăria SFML. Aceasta se poate găsi pe site-ul oficial: https://www.sfml-dev.org/download.php

Figură 1 Accesarea site-ului

Figure 2 Selectarea versiunii alături de compilatorul specific sistemului de operare

Această librărie cuprinde o multitudine de clase și metode speciale pentru randarea imaginilor, crearea ferestrelor, folosirea fișierelor audio, etc.

În funcție de software-ul pe care dorim să rulăm aplicația vom alege librăria corespunzătoare.

La crearea proiectului va fi necesară specificarea căilor folderelor lib, include și bin în setările IDE-ului pentru găsirea componentelor. Procesul explicit se poate găsi la pagina 12 Ssau:

Tutorial pentru setup: https://www.youtube.com/watch?v=4fcTqmT0Hhg&ab_channel=Tuffle

După instalarea librăriei vom crea proiectul în C++, în care vom include <SFML/Graphics.hpp> pentru partea grafică a jocului și <SFML/Audio.hpp> pentru partea audio.

Desigur, pe lângă aceste headere vom adăuga și iostream, array, vector și random pentru operații și prelucrări ulterioare.

În primul rând, pentru randarea jocului este nevoie de o fereastră pentru care trebuie să setăm lățimea și înălțimea. De asemenea trebuie să setăm o înălțime pentru Ground, o viteză cu

```
const unsigned int windowSize_x = 1000;
const unsigned int windowSize_y = 500;
const unsigned int groundOffset = windowSize_y - 150.f;
int gameSpeed = 8;
bool playerDead = false;
bool playDeadSound = false;
```

care se vor derula cadrele pe secundă și două condiții ca jocul să ruleze: playerDead = false și playDeadSound=false.

Figură 2 Inițializarea aplicației

Vom începe cu clasa Fps_s în care vom seta proprietățile fontului, a textului și a timpului. Vom afișa pe ecran numărul de cadre pe secundă în colțul din stânga sus cu metoda drawTo.

În clasa SoundManager vom inițializa trei buffere de tip SoundBuffer și trei instanțe de tip Sound aferente clasei sf. În fiecare din buffere vom atribui câte un sunet pentru acțiunea specifică iar ulterior vom asocia instanțelor de tip Sound bufferele corespondente.

```
dieBuffer.loadFromFile("rsrc/Sounds/die.wav");
jumpBuffer.loadFromFile("rsrc/Sounds/jump.wav");
pointBuffer.loadFromFile("rsrc/Sounds/point.wav");

dieSound.setBuffer(dieBuffer);
jumpSound.setBuffer(jumpBuffer);
pointSound.setBuffer(pointBuffer);
```

Figură 3 Asocierea sunetelor pentru acțiuni

În clasa Ground vom seta în primul rând textura ce se află în folderul „rsrc” – GroundImage.png și îi setăm înălțimea astfel încât să fie în partea de jos a ferestrei. Desigur, jocul este dinamic iar în timpul mișcării va trebui să actualizăm poziția Pământului atât timp cât playerDead=false. În cazul în care personajul va pierde, imaginea va rămâne statică, înafară de norii care vor continua să se miște. De asemenea ne va trebui o metodă de a reseta imaginea (void reset ()) în cazul în care jucătorul dorește să înceapă din nou jocul.

În clasa Obstacle vom asocia textura corespunzătoare, alături de marginile acestuia prin proprietatea clasei sf :: FloatRect. Obstacolul se va afla pe pământ.

În clasa Obstacles vom crea o listă de obiecte de tip Obstacle. Acestor obiecte le vom oferi variații ale texturilor pentru o diversitate a jocului. Acestea se vor afla în locuri aleatorii pe pământ, funcție implementată prin metoda „update” .

În clasa Dino vom avea toate proprietățile necesare precum: poziția, mișcarea, marginile pentru coliziuni, sunetele specifice, etc. Cât timp acesta este în viață, textura acestuia va fi dinamică, pentru a crea o animație de mișcare.

În clasa Scores vom avea scorul maxim, scorul precedent (dacă este cazul), scorul curent și diferența până la scorul maxim. Scorul curect se actualizează începând cu valoarea 0 cât timp playerDead = false. Când playerDead = true se va compara scorul curent, cu scorul precedent și cu scorul maxim pentru o ulterioară modificare a scorului maxim, dacă este cazul.

În clasa RestartButton inițializăm textura butonului pentru restart a jocului dacă jucătorul a pierdut și îi setăm poziția în fereastră.

În clasa Clouds vom inițializa mai întâi textura, setăm poziția verticală iar ne asigurăm că viteza cu care se vor deplasa norii să fie mai mică decât cea a pământului pentru o redare mai apropiată de realitate a efectului de depărtare.

Clasa GameState putem spune că este motorul aplicației. Aceasta se ocupă de crearea cadrului. În această clasă instanțiem obiecte ale celorlalte clase care fac parte din cadru. Această clasă este cea care pornește aplicația odată cu apelarea în funcția main.

În funcția main adăugăm ultimele detalii, precum setarea tagului de nume a aplicației, setarea dimensiunilor ferestrei, instanțierea clasei GameState și rularea aplicației.

4. TEHNOLOGII FOLOSITE – LIBRĂRIA SFML

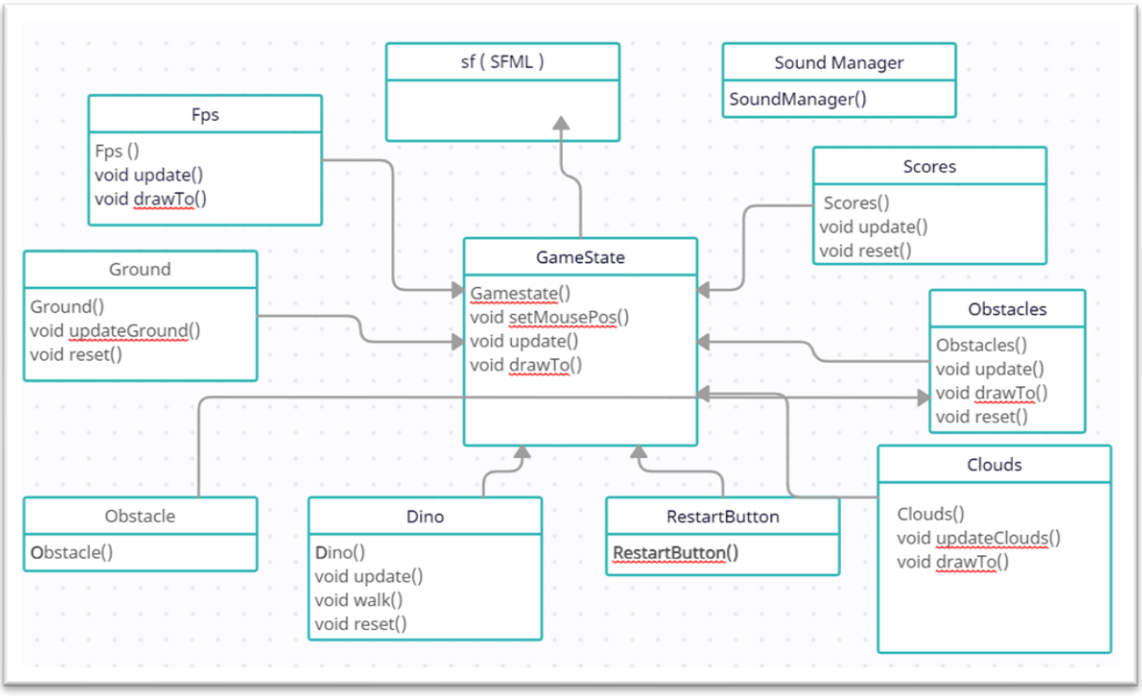
SFML (Simple and Fast Multimedia Library) este o bibliotecă open-source, gratuită și multiplatformă de programare pentru dezvoltarea de aplicații multimedia, în special pentru jocuri video și alte aplicații interactive.

SFML oferă o interfață simplă și eficientă pentru a accesa funcționalitățile multimedia ale unui sistem de operare, cum ar fi redarea audio și video, gestionarea ferestrelor și evenimentelor de la tastatură și mouse. Aceasta facilitează crearea de jocuri și aplicații interactive cu performanțe bune și o înaltă calitate vizuală și audio.

SFML este disponibilă pentru mai multe limbaje de programare, inclusiv C++, C#, Python și Ruby, ceea ce face posibilă dezvoltarea de aplicații pentru o varietate de platforme, inclusiv Windows, Linux, macOS și Android.

În concluzie, SFML este o bibliotecă utilă pentru programatorii care doresc să dezvolte jocuri și aplicații interactive cu interfață grafică, oferind o modalitate simplă și eficientă de a accesa funcționalitățile multimedia ale unui sistem de operare. De asemenea mulțumită IDE-ului Visual Studio 2022 Community Edition a fost posibilă crearea acestui proiect într-un mod mult mai eficient.

5. SCHEMA BLOC DE CLASE



Figură 4 Ierarhizarea claselor

6. FORMATUL DATELOR I/O

Aplicația pe care am creat-o are o structură I/O relativ simplă. Pentru începerea jocului este necesară apăsarea unui click de mouse iar pentru a face caracterul să sară este necesară apăsarea tastei “space”.

Datele de ieșire se afișează la nivel grafic, ca de exemplu afișarea scorului maxim, care se afișează în permanență și se schimbă de fiecare dată când jucătorul pierde, dacă a fost depășit.

7. STUDIU DE CAZ (LIMITĂRI/BUGS)

- În primul rând, această aplicație are, ca orice altă aplicație limitările sale. De exemplu în cazul în care jocul este rulat la 30 fps pe un laptop (când procesorul este în mod idle pentru economisirea bateriei) evenimentele se vor derula mult mai lent, întrucât evenimentele depind de numărul de cadre pe secundă.

- Marginile obstacolelor nu sunt exacte, având o formă de dreptunghi, rezultând la o posibilă pierdere chiar dacă personajul nu a atins obstacolul în mod fizic.

- În momentul în care se execută un restart al jocului din tasta “space” personajul va sări la început de rundă.

- În momentul în care jucătorul ajunge la scorul de aproximativ 9000 viteza de derulare a jocului poate afecta calitatea gameplay-ului.

8. MANUAL DE UTILIZARE

1. Introducere și prezentare

Bun venit în lumea jocului Dino Run. Acest joc captivant îți oferă o experiență unică de divertisment și aventură. Explorează lumea preistorică și evită obstacolele pentru a obține un scor cât mai mare!

2. Cerințe de sistem

- Sistem de operare: Windows 7/8/10/11 (versiune x64 biți)

- Procesor: Intel Pentium 4 sau echivalent minim
- Memorie RAM: 2 GB sau mai mult
- Spațiu de stocare disponibil: 20 MB
- Placă video: compatibilă DirectX 9 sau ulterior
- Sunet: Dispozitiv audio compatibil

3. Instalare și configurare

Pentru a instala jocul “Dino Run”, urmați acești pași:

a) Descărcați proiectul de pe repository-ul de pe GitHub:

https://github.com/OlivianVarareanu/Dino-Run-PROIECT-OOP

b) Pentru accesarea aplicației va fi nevoie de deschiderea fișierului “Proiect.sln” cu Visual Studio Community Edition 2022 sau versiune ulterioară.

c) După deschidere va fi necesară setarea din Visual Studio a locațiilor specifice ale librăriei FSML, care se află în interiorul aceluiași folder.

d) În tab-ul **Solution Explorer** din Visual Studio selectăm fișierul Proiect, după care apăsăm **Click-dreapta -> Properties**

e) În fereastra **Proiect Property Pages**, în partea stanga se vor afla mai multe Proprietăți. În **Configuration Properties -> General** ne vom asigura că **Windows SDK Version** este **versiunea 10.0** și **Platform Toolset** este **Vistual Studio 2022 (v143).**

f) În **Configuration Properties -> Debugging -> Environment** vom scrie “ **PATH=** “ **,** urmat de path-ul corespunzător folderului **bin.** Exemplu: **PATH=D:\Proiect\SFML-2.5.1\bin** , după care apăsăm tasta enter și butonul **Apply**.

g) În **C/C++ -> General** la **Additional Inlcude Directories** vom seta path-ul corespunzător folderului **include** din librăria **SFML-2.5.1**. Exemplu: D:\Proiect\SFML-2.5.1\include, după care apăsăm tasta enter și butonul **Apply**.

h) În **Linker -> General** la **Additional Library Directories** vom seta path-ul corespunzătpr folderului **lib** din librăria **SFML-2.5.1**. Exemplu: D:\Proiect\SFML=2.5.1\lib, după care apăsăm tasta enter și butonul **Apply**.

i) În **Linker -> Input** la **Additional Dependencies** vom seta ‘’ **sfml-graphics-d.lib;sfml-window-d.lib;sfml-audio-d.lib;sfml-network-d.lib;sfml-system-d.lib;%(AdditionalDependencies) ‘’** , după care apăsăm tasta enter și butonul **Apply**.

j) După ce am setat dependențele față de această librărie putem închide fereastra de proprietăți și rula aplicația prin butonul **Local Windows Debugger**.

k) Enjoy!

4. Interfață utilizator

- Interfața utilizator a jocului Dino Run este intuitivă și ușor de utilizat. Elementele principale includ:

- Ecranul jocului, scorul curent, scorul maxim, numărul de cadre pe secundă și starea personajului .

5. Obiectiv și Gameplay

- Obiectivul jocului este să eviți obstacolele pentru a obține un scor cât mai mare.

9. CAPTURI DE ECRAN

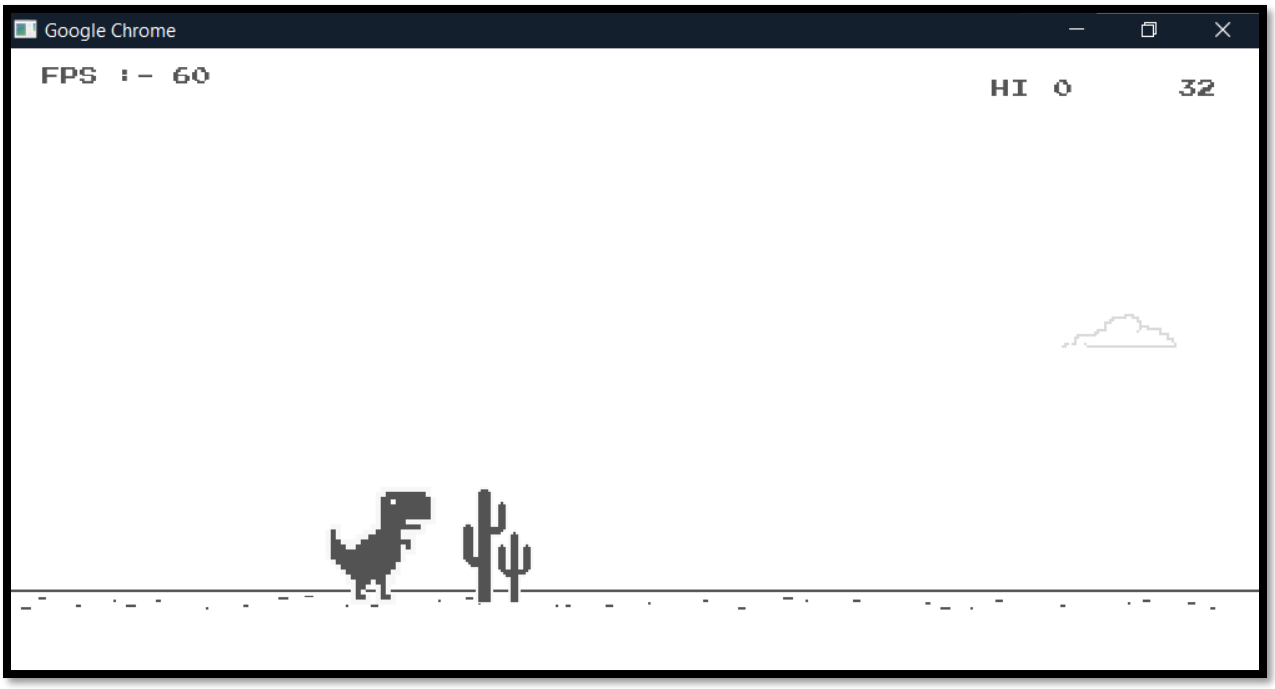


Figure 1 Exemplu de gameplay

Figure 4 Exemplu în care jucătorul a pierdut

10. BIBLIOGRAFIE

- https://www.sfml-dev.org/documentation/2.5.1/modules.php
- https://www.sfml-dev.org/documentation/2.5.1/annotated.php
- https://www.youtube.com/playlist?list=PL6xSOsbVA1eaJnHo_O6uB4qU8LZWzzKdo
- https://en.wikipedia.org/wiki/Dinosaur_Game
- https://www.sfml-dev.org/tutorials/2.5/