Colegiul Tehnic INFOEL

*Bistrita*

**Proiect pentru atestarea competențelor profesionale la informatică**

**Profil: real**

Specializarea: Matematică-informatică, intensiv informatică

Anul 2021

Joc video: Aurora sees fire

**Autor: Coordonator: prof.**

**Uifelean Aurora Sofia Bîz Maria Felicia**

**Bistrița, 2021**

**Cuprins**

**1. MOTIVAȚIA ALEGERII TEMEI………………………………………….…..……..………….3**

**2. INTRODUCERE ÎN .NET ……………………………………….………………………………..3**

2.1. CONCEPTE DE BAZĂ ALE PROGRAMĂRII VIZUALE….…………….……………..3

2.2. ELEMENTELE POO ÎN CONTEXT VIZUAL……..………….….………....……..…….3

* FEREASTRA TOOLBOX…………………………………….….…………………….4
* FEREASTRA SOLUTION EXPLORER………………………………….…………...4
* CONTROALE………………………………………….………………………………4

**3. DESCRIEREA JOCULUI………………………………………….…………….……………..…6**

**4. MENIU………………………………………………………….………….………………………..6**

4.1. DESPRE………………………………………………….………………………….……..9

4.2. INSTRUCȚIUNI…..………………………………………………………….…..………10

4.3. AUTENTIFICARE UTILIZATORI ……………………………….…………………….10

4.4. AUTENTIFICARE ADMINISTRATORI………………………………………………..12

4.5. ÎNREGISTRARE…………………………………………………………………………13

4.6. PANOU CONTROL UTILIZATORI…………………………………………………….14

**5. JOC…………………………………………………………………………………………………16**

* BACKGROUND………………………………………………………………………………17
* PROTAGONIST………………………………………………………………………………18
* GLOANȚELE PROTAGONISTULUI………………………………………………………..19
* INAMICI.……………………………………………………………………………………...20
* PROIECTILELE INAMICILOR……………………………………………………………...21
* COLIZIUNE…….……………………………………………………………………………..22
* SUNETE..……………………………………………………………………………………...24
* PAUZĂ……………………………………………………………………………………...…25

**6. BIBLIOGRAFIE………………………..…………………………………………………………25**

**1. MOTIVAȚIA ALEGERII TEMEI**

Tema aleasă de mine este un joc video în 2D, jocul fiind unul de tipul shoot ‘em up. Shoot ‘em up sau STG (abrevierea japoneză comună pentru „jocuri de tragere”), este un joc în care protagonistul trage într-un număr mare de inamici în timp ce evită atacurile acestora.

Am ales această temă pentru a ajuta copiii să-și dezvolte viteza de racție, viteza de procesare a informației și procesarea imaginilor vizuale. De asemenea, m-am gândit că această aplicație poate fi de folos spitalelor echipate cu stații de jocuri care pot avea un impact pozitiv asupra recuperării pacienților. Pacienții pot găsi anumite distracții și bucurii prin acțiunea de a se juca.

**2. INTRODUCERE ÎN .NET**

NET este un cadru (*Framework*) de dezvoltare software unitară care permite realizarea, distribuirea şi rularea aplicaţiilor desktop Windows şi aplicaţiilor WEB.

Tehnologia .NET pune laolaltă mai multe tehnologii (ASP, XML, OOP, SOAP, WDSL, UDDI) şi limbaje de programare (VB, C++, C#, J#), punând la dispoziția noastră posibilitatea de a dezvolta diferite aplicații.

2.1. CONCEPTE DE BAZĂ ALE PROGRAMĂRII VIZUALE

Programarea vizuală reprezintă crearea unei aplicații prin operare directă asupra unui set de elemente grafice. Această operare are ca efect scrierea automată a unor secvenţe de program, secvenţe care, împreună cu secvenţele scrise de textual de utilizator vor forma programul.

Realizarea unei aplicaţii vizuale nu ține doar de aranjarea elementelor grafice într-o fereastră. Fiecare element grafic are proprietățile lui, iar cu ajutorul acestor proprietăți și a secvențelor de cod putem într-adevăr să numim că ceea ce facem este programare vizuală orientată pe obiecte.

2.2. ELEMENTELE POO ÎN CONTEXT VIZUAL

* Fereastra Toolbox
* Fereastra Windows Forms
* Fereastra Solution Explorer
* Fereastra Listei de erori
* Fereastra Properties
* Bara de meniuri
* Bara de unelte

Repartizarea elementelor de programare vizuală orientată pe obiecte în Visual Studio 2019

Fereastra Windows Forms

Designer în care s-a creat

Form1

Fereastra Solution

Explorer

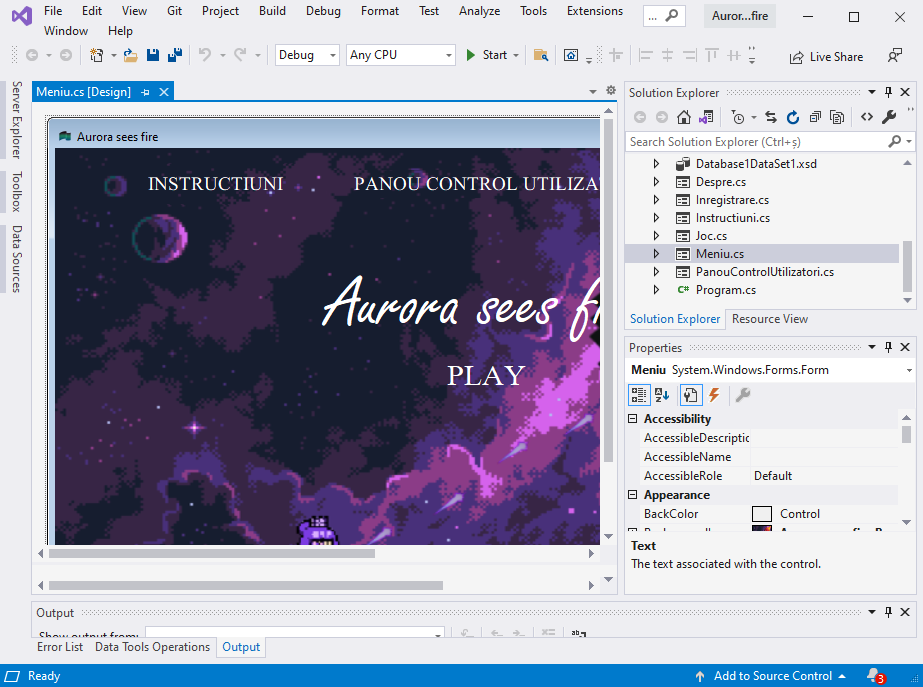
Fereastra Properties

Fereastra Toolbox

Fereastra Listei de erori

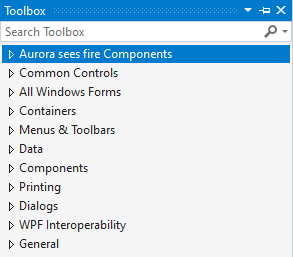
Bara de meniuri

Bara de unelte



* **FEREASTRA TOOLBOX**

Opțiunile din fereastra toolbox se deschid apăsând semnul săgeată din faţă. De exemplu, dacă se deschide **Common Controls** în fereastră apar controale mai des folosite. Orice control poate fi adus în fereastra în care lucrăm prin dublu clic pe respectivul control, sau prin drag and drop în fereastră.

****

* **FEREASTRA SOLUTION EXPLORER**

Fereastra Solution Explorer, din partea dreaptă se referă, printre altele la ferestra Designer sau la fereastra în care utilizatorul va scrie propriul cod.

În cazul în care fereastra Designer este închisă, putem apela la opţiunea Open şi va reapărea în fereastra centrală. Dacă dorim să vedem codul, apăsăm pe opţiunea View Code, iar în fereastra principală se va deschide, încă o ferestră corespunzătoare codului dorit.

* **CONTROALE**

|  |  |  |
| --- | --- | --- |
| **Funcţia controlului** | **Numele**  **controlului** | **Descriere** |
| buton | **Button** | Sunt folosite pentru a executa o secvenţă de instrucţiuni în momentul activării lor de către utilizator |
| casetă de validare | **CheckBox** | Oferă utilizatorului opţiunile: da/nu sau include/exclude |
| etichetă | **Label** | Sunt folosite pentru afişarea etichetelor de text, şi a pentru a eticheta controalele. |
| imagine | **PictureBox** | Este folosit pentru adăugarea imaginilor sau a altor resurse de tip bitmap. |
| casetă de text | **TextBox** | Este utilizat pentru afişarea textului generat de o aplicaţie sau pentru a primi datele introduse de la tastatură de către utilizator. |

**3. DESCRIEREA JOCULUI**

Jocul este unul de tipul shoot ‘em up. Shoot ‘em up sau STG (abrevierea japoneză comună pentru „jocuri de tragere”), este un joc în care protagonistul trage într-un număr mare de inamici în timp ce evită atacurile acestora. Jucătorul trebuie să se bazeze în primul rând pe timpii de reacție pentru a câștiga cât mai multe puncte, sau pentru a câștiga jocul.

În Aurora sees fire este vorba despre prințesa Aurora care călătorește prin spațiu și se întâlnește cu diverse comete de culori diferite care vor să-i fure energia pozitivă. Jucătorul trebuie să o ajute pe Aurora astfel încât să-și protejeze energia pozitivă și să învingă cometele.

Aplicația este compusă din jocul în sine și din baza de date pentru utilizatori. Utilizatorii se pot autentifica dacă au cont, iar dacă nu au își pot crea un cont nou. Administratorii au de asemenea un spațiu unde se pot autentifica. Administratorii au acces la baza de date cu utilizatorii și pot modifica date din aceasta, precum: adăugare de utilizatori, ștergerea utilizatorilor, schimbarea parolei sau a funcției unui utilizator, etc.

**4. MENIU**

Aplicația se deschide cu o fereastră ”Meniu” ce conține meniul jocului. Background-ul este reprezentat de o imagine în care Aurora trage într-o cometă care vrea să-i fure energia pozitivă. În această fereastră sunt 7 butoane: „instructiuni” (deschide formul cu instrucțiunile jocului), „despre” (deschide formul cu descrierea jocului), „autentificare utilizatori” (deschide panoul de autentificare pentru utilizatori), „autentificare administratori” (deschide panoul de autentificare pentru administratori), „panou control utilizatori” (apare doar daca utilizatorul autentificat este administrator si deschide panoul de administrare al utilizatorilor), „play” (deschide formul cu jocul) și „iesire” (închide aplicația). Pe lângă butoanele menționate mai sus se mai află Label-ul „titlu” care conține numele jocului.

Codurile evenimentelor în urma cărora prin apăsarea butoanelor instruciuni și despre se deschid formurile Instructiuni și respectiv Despre:

private void instructiuni\_Click(object sender, EventArgs e)

{

Instructiuni f = new Instructiuni();

this.Location = new Point(100, 50);

f.Show();

}

private void despre\_Click(object sender, EventArgs e)

{

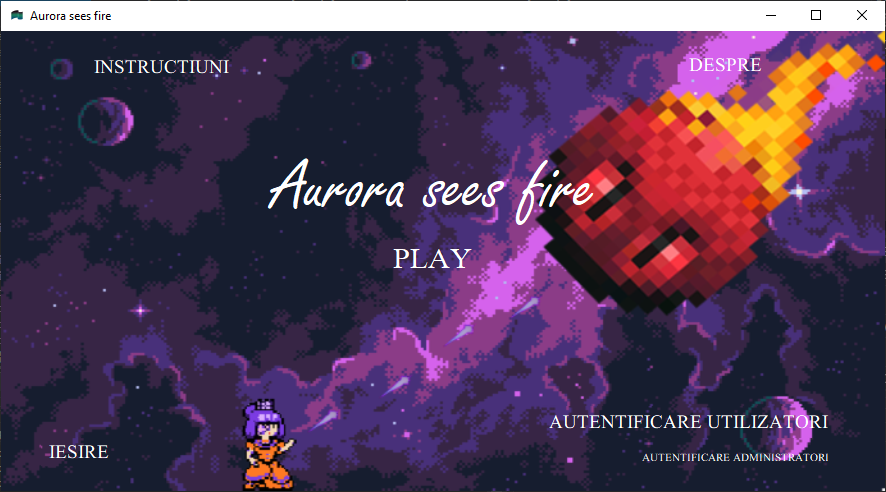
Despre f = new Despre();

this.Location = new Point(100, 50);

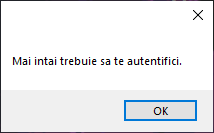
f.Show();

}

Formul Meniu care reprezintă meniul jocului:



Pentru ca utilizatorul să se poată juca, mai întâi trebuie să se autentifice, altfel va apărea un MessageBox cu mesajul: „Mai intai trebuie sa te autentifici.”. MessageBox-ul și codul evenimentului:



private void play\_Click(object sender, EventArgs e)

{

if (ok == 1)

{

login = 1;

Joc.Joc f = new Joc.Joc();

this.Location = new Point(100, 50);

f.ShowDialog();

punctaj = f.scor;

}

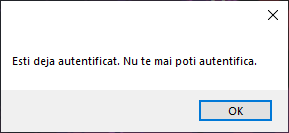
else

MessageBox.Show("Mai intai trebuie sa te autentifici.");

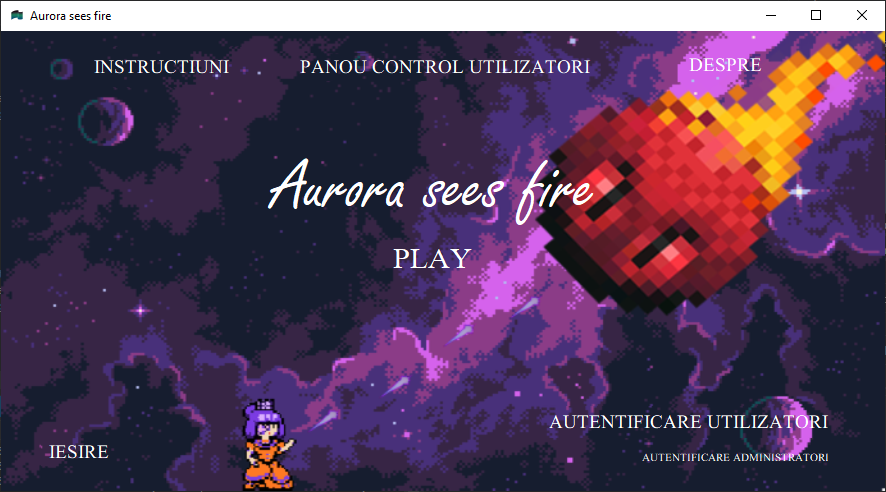
}

Acest lucru l-am făcut cu ajutorul a 3 variabile: ok care inițial este 0, login care inițial este 0 și admin care inițial este 0. Când variabila login este egală cu 0 înseamnă că utilizatorul nu s-a autentificat și se poate autentifica. După ce se autentifică, dacă este doar utilizator atunci ok și login devin 1, iar dacă este administrator atunci și admin devine 1.

Dacă variabila ok este egală cu 1, apăsând pe butonul play se deschide fereastra „Joc”. Odată autentificat, utilizatorul nu se mai poate atentifica, deoarece butoanele deschid panourile de autentificare doar dacă login este egal cu 0. Dacă utilizatorul încearcă să se mai autentifice atunci apare MessageBox-ul următor:



Dacă variabila admin este egală cu 1, atunci utilizatorul are funcție de administrator, iar butonul „panou\_control\_utilizatori” devine vizibil (inițial era invizibil).



**4.1. DESPRE**

Fereastra „Despre” se deschide în momentul apăsării butonului „despre” din fereastra „Meniu”. Această fereastră este constituită din descrierea jocului inserată în Label-uri, alături de un PictureBox cu poza protagonistei jocului numită Aurora și un buton pentru întoarcerea în meniu.



De asemenea, am inserat o scurtă descriere a mea și am oferit credite pentru muzica de fundal din timpul jocului și pentru protagonistă.



Poza de mai sus este un pixel art. Am ținut să modific pixel art-ul în Adobe Photoshop 2021 deoarece am vrut să am o implicare și la nivel de artă digitală pentru a personaliza mai mult această aplicație. Caracterul trebuie să fie cât mai user-friendly.

Această poză este un vector (are fundalul transparent), ceea ce m-a ajutat mai târziu când a trebuit să pun aceeași poză în joc.

**4.2. INSTRUCȚIUNI**

Fereastra „Instricțiuni” se deschide în momentul apăsării butonului „instrucțiuni” din fereastra „Meniu”. Fereastra „Instrucțiuni” are următoarele elemente din Toolbox: 5 PictureBox-uri în care am pus pozele unor taste specifice pentru a juca, Label-uri pentru descrierea acțiunilor în urma apăsări tastelor și un Button pentru întoarcerea în meniu.



**4.3. AUTENTIFICARE UTILIZATORI**

Pentru a se deschide fereastra „AutentificareUtilizatori” trebuie apăsat Button-ul „autentificare\_utilizatori” din fereastra „Meniu”. Această fereastră este deschisă cu opțiunea ShowDialog, însemnând că fereastra se închide doar când utilizatorul se loghează, mai ales că am dezactivat butoanele de minimize, maximize și close ale ferestrei.



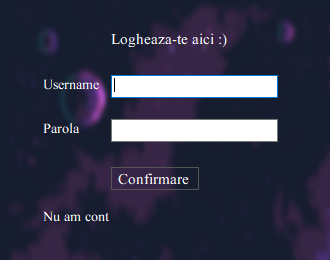
Secvență de cod din fereastra „Meniu” în care am introdus apelul ShowDialog:

Autentificare f2 = new Autentificare();

f2.ShowDialog();

Controalele utilizate în fereastră sunt: un GroupBox pentru înglobarea elementelor și design, 4 Label-uri (unul pentru titlul ferestrei, două numite „username” și „parolă” și unul pentru opțiuniea „Nu am cont”), două TextBox-uri (unul pentru scrierea username-ului și celălalt pentru parolă, pentru logare) și un Button pentru confirmarea datelor de logare.

Captură de ecran cu fereastra „AutentificareUtilizatori”:



Am creat o bază de date cu un tabel „Utilizatori” care să mă ajute să rețin datele utilizaorilor. De asemenea, am inițializat o variabilă idu pentru a memora id-ul utilizatorului. Asta mă va ajuta mai târziu în reținerea scorului în baza de date, id-ul fiind trimis în meniu.

Codul SQL care verifică dacă corespund în baza de date username-ul cu parola introduse de utilizator în fereastra de autentificare a utilizatorilor:

SELECT COUNT(\*) FROM Utilizatori

WHERE username = @username and parola = @parola

Codurile evenimentului Click al Button-ului „confirmare\_logare”:

private void confirmare\_logare\_Click(object sender, EventArgs e)

{

string username = "", parola = "";

if (textBox1.Text != "" && textBox2.Text != "")

{

username = textBox1.Text;

parola = textBox2.Text;

if (utilizatoriTableAdapter.ScalarQueryLogare(username, parola) != 0)

{

MessageBox.Show("Bine ai venit, " + textBox1.Text + "!");

idu = utilizatoriTableAdapter.ScalarQueryGasireId(username, parola).ToString();

this.Close();

}

else

MessageBox.Show("Date de autentificare gresite");

}

else

MessageBox.Show("Introduceti username si parola");

}

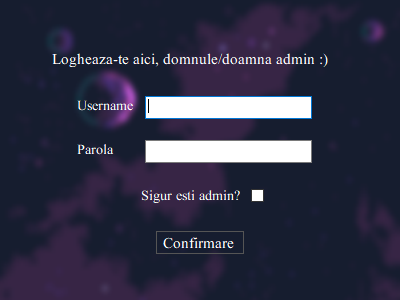
**4.4. AUTENTIFICARE ADMINISTRATORI**

Fereastra „AutentificareAdministratori”, la fel ca și celălalte de până acum, se deschide apăsând Button-ul „autentificare\_administratori”. De asemenea, are aceeași proprietate ca fereastra „AutentificareUtilizatori”: se închide odată ce datele completate sunt corecte.

Fată de cealaltă autentificare, la autentificarea pentru administratori am adăugat un CheckBox alături de care este întrebarea „Sigur ești admin?”.

Există 6 situații posibile la această fereastră și am să le enumăr mai jos alături de mesajul care apare în urma selectări Button-ului „confirmare”:

1. Datele introduse sunt incomplete: „Introduceți username și parolă!”.
2. Username-ul și parola nu corespund: „Date de autentificare greșite!”.
3. Utilizatorul nu este admin și nu bifează CheckBox-ul: „Nu ești administrator!”.
4. Utilizatorul nu este admin și bifează CheckBox-ul: „Nu ești administrator!”.
5. Utilizatorul este admin și nu bifează CheckBox-ul: „Ești administrator, dar trebuie să bifezi căsuța!”.
6. Utilizatorul este admin și bifează CheckBox-ul: „Bine ai venit, admin ” + textbox1.Text + „!” (în textbox1 se află username-ul administratorului și îl preia cu proprietatea .Text.



Interogarea pe care am folosit-o pentru a verifica dacă un utilizator are funția de administrator sau nu am numit-o ScalarQueryAdmin, construită cu ajutorul limbajului SQL și returnează o singură valoare. Codul este următorul:

SELECT COUNT(\*) AS Expr1

FROM Utilizatori

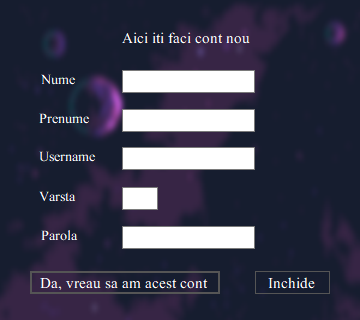
WHERE (username = @username) AND (parola = @parola) AND (admin = @admin)

**4.5. ÎNREGISTRARE**

Fereastra „Înregistrare” se deschide la apăsarea Label-ului „nu\_am\_cont” din fereastra „AutentificareUtilizator”. Această fereastră pune la dispoziția utilizatorului să își facă un cont în aplicație. În eventualitatea în care utilizatorul are cont și a apăsat din greșeală pe Label-ul „nu\_am\_cont”, există în fereastra de înregistrare Button-ul „inchide” care iese de fereastra actuală și revine la cea de autentificare a utilizatorilor.

Crearea unui cont constă în memorarea datelor introduse de utilizator în baza de date. Datele care se pot memora sunt următoarele: nume, prenume, username, vârsta și parola. Un simplu utilizator poate avea funcția de administrator doar dacă un administrator intră în baza de date și îi modifică funcția.

În urma completării corecte și totale a datelor se apasă pe Button-ul „cont\_creat”, care are textul „Da, vreau să am acest cont” pentru a salva datele contului în baza de date.



Există 3 situații posibile la această fereastră și am să le enumăr mai jos alături de mesajul care apare în urma selectări Button-ului „cont\_creat”:

1. Nu au fost completate toate TextBox-urile: „Toate câmpurile sunt obligatorii!”.
2. Nu s-au introdus corect datele (de exemplu, la vârstă s-a scris una sau mai multe litere): „A apărut o eroare la adăugarea contului!”.
3. Datele sunt corecte și pot fi reținute în baza de date: „Contul a fost creat cu succes!”.

Odată cu crearea contului și memorarea datelor în baza de date, în tabelul bazei de date pe coloana „scor” se inserează automat baloarea 0.

Interogarea pe care am folosit-o în această fereastră de înregistrare a utilizatorilor am numit-o InsertQueryUtilizatori, construită cu ajutorul limbajului SQL și este de tipul INSERT, deoarece trebuie să insereze datele luate din TextBox-uri în tabelul „Utilizatori” al bazei de date „Database1DataSet”. Codul acestei interogări este următorul:

INSERT INTO Utilizatori

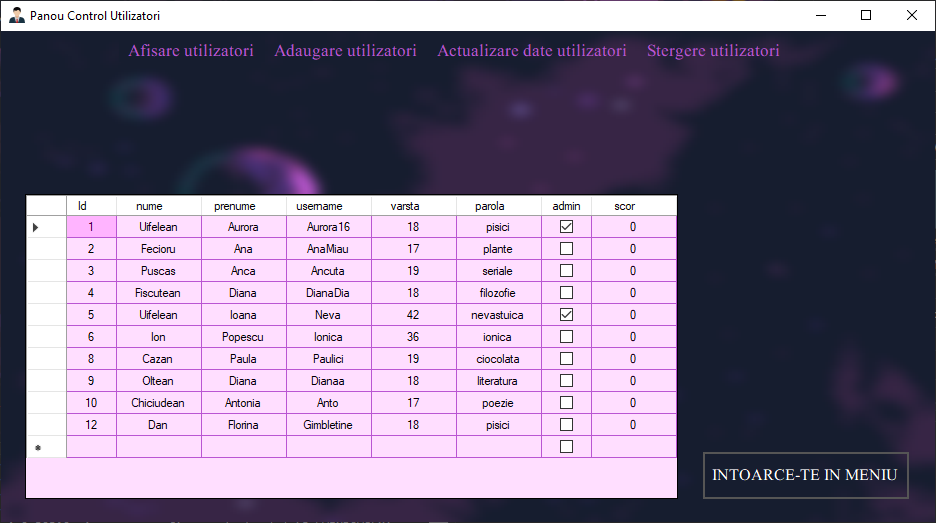
(nume, prenume, username, varsta, parola, scor)

VALUES (@nume,@prenume,@username,@varsta,@parola, 0);

SELECT Id, nume, prenume, username, varsta, parola, scor FROM Utilizatori WHERE (Id = SCOPE\_IDENTITY())

**4.5. PANOU CONTROL UTILIZATORI**

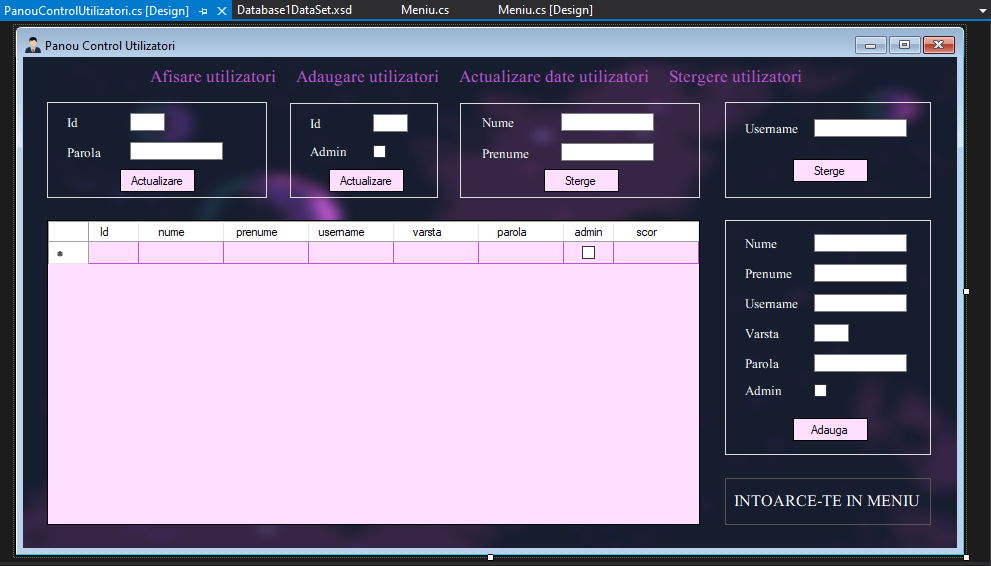
Fereastra „PanouControlUtilizatori” se deschide la apăsarea Button-ului „panou\_control\_utilizatori” din fereastra „PanouControlUtilizatori”. Această fereastră dispune de o interfață simplă și ușor de folosit de către administratori pentru a edita utilizatori.



În fereastra curentă am inserat următoarele elemente: tabelul „Utilizatori”, un MenuStrip, 5 GroupBox-uri, 12 Label-uri, două CheckBox-uri și 6 Button-uri. În urma acțiunii click ale unor Button-uri din MenuStrip, apar GroupBox-uri specifice pentru a adăuga utilizatori, pentru a șterge și pentru a actualiza date.

De asemenea, administratorul are posibilitatea de a vedea utilizatorii din tabelul „Utilizatori” în ordine alfabetică după nume, după username, doar administratorii și doar utilizatorii. Cu ajutorul Button-ului din dreapta jos administratorul se poate întoarce în meniu.

Acestea sunt GroupBox-urile care apar în urma apăsării Button-urilor din MenuStrip:



Codul pentru ștergerea unui utilizator după nume și prenume:

private void button3\_Click(object sender, EventArgs e)

{

string nume = "", prenume = "";

if(textBox6.Text != "" && textBox7.Text != "")

{

nume = textBox6.Text;

prenume = textBox7.Text;

utilizatoriTableAdapter1.DeleteQueryStergeUtilizatoriDupaNumePrenume

(nume, prenume);

}

this.utilizatoriTableAdapter1.Fill(this.database1DataSet1.Utilizatori);

}

Interogarea pe care am folosit-o am numit-o DeleteQueryStergeUtilizatoriDupaNumePrenume, construită cu ajutorul limbajului SQL și este de tipul DELETE, deoarece șterge date din tabel. Tabelul este tot cel de până acum, adică „Utilizatori”, iar baza de date este Data1DataSet1. Codul acestei interogări este următorul:

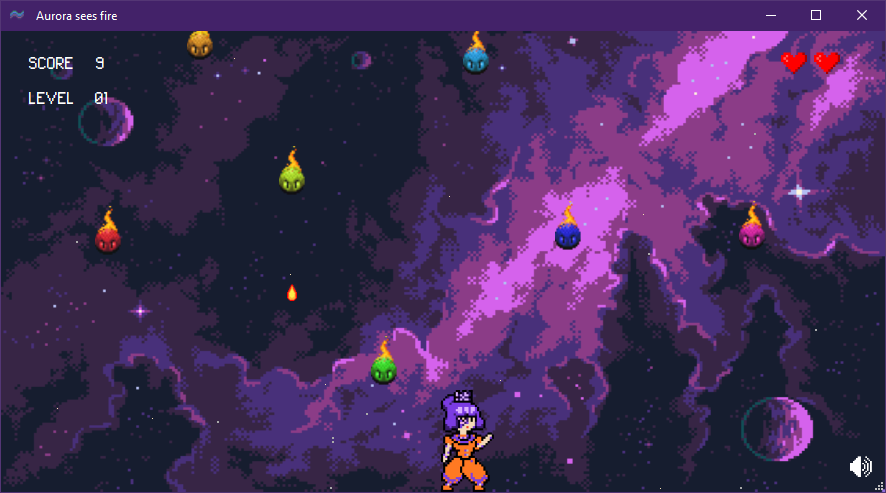
DELETE FROM [dbo].[Utilizatori]

WHERE ([nume] = @Original\_nume) AND ([prenume] = @Original\_prenume)

**5. JOC**

După cum am scris și mai sus, jocul este unul de tipul shoot ‘em up. Shoot ‘em up sau STG (abrevierea japoneză comună pentru „jocuri de tragere”), este un joc în care protagonistul trage într-un număr mare de inamici în timp ce evită atacurile acestora. În cazul acestui joc, prințesa Aurora trebuie să se ferească de atacurile inamiciilor și de aceștia în timp ce trage cu o rază de energie în ei.

La deschiderea aplicației am făcut o animație de aproximativ 40 de secunde, animație în care se prezintă povestea prințesei Aurora și de ce este urmărită de entitățile din galaxia Andromeda.



Poziția inamiciilor este aleatorie, la fel ca inamicii care trag, iar în colțul din dreapta sus sunt reprezentate cele 3 vieți. Jucătorul pierde o viață dacă:

1. Este lovit de bila de foc lansată de un inamic.
2. Se ciocnește de un inamic.

De asemenea, după cum am menționat și în instrucțiuni, volumul jocului este reglabil. În primă fază volumul este maxim, iar la apăsarea tastei „M” volumul scade. Sunt 4 trepte de sunet: tare, mediu, încet și mut. Eu am făcut pixel-art-urile pentru sunet, inimi și energia cu care trage prințesa Aurora. Programul în care am făcut aceste pixel-art-uri se numește Aseprite.

Sunet:    

Inimi:   

Energie: 

În colțul din stânga sus se află nivelul și scorul jucătorului. Scorul reprezintă numărul de inamici distruși de prințesa Aurora. Odată la 25 de puncte de scor crește nivelul, nivelul inițial fiind „Level 01”.

Pentru a salva id-ul utilizatorului/administratorului și scorul, am inițializat public 3 variabile: punctaj, idutilizator și idadministrator, pe care le-am transmis în antetul ferestrei „Joc”. În acest mod scorul se actualizează în baza de date în funcție de id-ul utilizatorului/administratorului. Codul este următorul:

public Joc(int punctaj, string idutilizator, string idadministrator)

{

InitializeComponent();

idut = idutilizator;

idad = idadministrator;

}

Practic, eu preiau scorul din joc și îl pun în baza de date în locul id-ului salvat la autentificare doar dacă scorul inițial este mai mare decât ultimul scor reținut în baza de date ultima dată (dacă utilizatorul este nou atunci scorul lui din baza de date va fi 0).

* BACKGROUND

Background-ul jocului este reprezentat de o imagină pixelată din spațiu cu nori de substanțe, tonurile fiind de un mov spre indigo deschis. De asemenea, pentru că prințesa Aurora călătorește prin spațiu, am ținut să fac un bacground cu stele care se miscă în jos, ca să dea senzația că Aurora zboară. Pentru asta am avut nevoie de un vector de 30 de PictureBox-uri pe care eu le-am împrțit în două categorii: stele mici și albe (X = 1 px și Y = 1 px) și stele gri și mari (X = 2 px și Y = 2 px):

stele\_de\_fundal = new PictureBox[30]; rnd = new Random();

for (int i = 0; i < stele\_de\_fundal.Length; i++)

{

stele\_de\_fundal[i] = new PictureBox();

stele\_de\_fundal[i].BorderStyle = BorderStyle.None;

stele\_de\_fundal[i].Location = new Point(rnd.Next(20, 885), rnd.Next(-10, 865));

if (i % 2 == 1)

{

stele\_de\_fundal[i].Size = new Size(1, 1);

stele\_de\_fundal[i].BackColor = Color.Wheat;

}

else

{

stele\_de\_fundal[i].Size = new Size(2, 2);

stele\_de\_fundal[i].BackColor = Color.DarkGray;

}

this.Controls.Add(stele\_de\_fundal[i]);

}

Mișcarea lor am făcut-o cu ajutorul unui Timer pe care l-am numit Miscare\_background (Interval: 10 milisecunde) și m-am ajutat de o variabilă viteza\_de\_background (inițial egală cu 4).

* PROTAGONIST

Protagonistul acestui joc este (evident) prințesa Aurora. Ea are o energie pozitivă unică în univers, motiv pentru care este urmărită de către entitățile din galaxia Andromeda, entități care seamănă cu niște asteroizi. Pixel-art-ul Aurorei l-am luat de pe internet, ulterior modificându-l deoarece am vrut să vin cu o amprentă personală și de asemenea am vrut să fac două animații: una cu Aurora când stă și respiră și una cu ea când merge.

Jucătorul se poate mișca de pe tastele săgeată stânga și săgeată dreapta. Acest lucru l-am făcut cu ajutorul a două Timere: Miscare\_Stanga și Miscare\_Dreapta și am mai folosit o variabilă pentru viteză: viteza\_jucator.

private void Miscare\_Stanga\_Tick(object sender, EventArgs e)

{

if (Jucator.Left > 10)

{

Jucator.Left -= viteza\_jucator;

st = 1;

dr = 0;

Jucator.Image = Image.FromFile(@"asserts\jucator-stanga.gif");

}

}

private void Miscare\_Dreapta\_Tick(object sender, EventArgs e)

{

if (Jucator.Left < 800)

{

Jucator.Left += viteza\_jucator;

st = 0;

dr = 1;

Jucator.Image = Image.FromFile(@"asserts\jucator-dreapta.gif");

}

}

Aceste evenimente determină mișcarea jucătorului la dreapta sau la stânga. Dacă jucătorul merge la stânga ajungi imaginea acestuia va fi un gif cu Aurora care merge la stânga, gif prealuat din folderul „asserts” unde se află pozele. La fel se întâmple și când jucătorul merge la dreapta, doar că atunci gif-ul va fi cu Aurora care merge la dreapta.

Jucătorul se poate mișca doar stânga – dreapta și mișcarea se face între pixelii 11 și 799 pe axa OX (poziția jucătorului este reținută permanent)

Codurile scrise în eventimentul KeyDown sunt executate în momentul tastei precizate și apăsate de la tastatură, iar evenimentul KeyUp execută codurile scrise în el în momentul ridicării degetelor de pe tastele precizate:

În KeyDown: În KeyUp:

if (e.KeyCode == Keys.Right) Miscare\_Dreapta.Stop();

{ Miscare\_Stanga.Stop();

Miscare\_Dreapta.Start();

}

if (e.KeyCode == Keys.Left)

{

Miscare\_Stanga.Start();

}

* GLOANȚELE PROTAGONISTULUI

Aurora trage cu energie, energie care se materializează și arată ca o rază de culoare roz cu mov. Jucătorul trage de pe space în inamici, iar aici de asemenea am avut nevoie de un Timer pe care l-am numit Miscare\_glont, având intervalul de 10 milisecunde. Tot aici am avut nevoie de o variabilă viteza\_glont. Acesta este evenimentul care o ajută pe Aurora să tragă:

private void Miscare\_glont\_Tick(object sender, EventArgs e)

{

for (int i = 0; i < gloante.Length; i++)

{

Coliziune();

if (gloante[i].Top > 0 && glontok == 1)

{

gloante[i].Visible = true;

gloante[i].Top -= viteza\_glont;

}

else

{

gloante[i].Visible = false;

if (dr == 1)

{

gloante[i].Location = new Point(Jucator.Location.X + 46, Jucator.Location.Y + 20);

}

if (st == 1)

{

gloante[i].Location = new Point(Jucator.Location.X + 1, Jucator.Location.Y + 20);

}

gloante[i].BringToFront();

glontok = 0;

sunetmuzicaglont = 0;

}

}

}

Locația din care pornește raza de energie cu care trage Aurora este în funcție de poziția protagonistei. Dacă Aurora merge în dreapta atunci raza pornește din Jucator.Location.X + 46, Jucator.Location.Y + 20, altfel pornește din Jucator.Location.X + 1, Jucator.Location.Y + 20.

* INAMICI

Inamicii protagonistei jocului sunt entități din galaxia Andromeda care seamănă cu niște comete/asteroizi în flăcări. Odată cu omorârea primului set de inamici, ceilalți inamici au poziții aleatorii. Imaginea unui astfel de inamic am luat-o de pe internet și ulterior am modificat-o în Adobe Photoshop 2021. Aceștia sunt toți inamicii:

Am declarat un șir de 8 PictureBox-uri cu aceleași proprietăți reprezentând inamicii.

inamici = new PictureBox[8];

for (int i = 0; i < inamici.Length; i++)

{

inamici[i] = new PictureBox();

inamici[i].Size = new Size(30, 50);

inamici[i].SizeMode = PictureBoxSizeMode.Zoom;

inamici[i].Location = new Point((i + 1) \* 92, rnd.Next(-30, -10));

inamici[i].BackColor = Color.Transparent;

inamici[i].BorderStyle = BorderStyle.None;

inamici[i].Visible = false;

this.Controls.Add(inamici[i]);

}

Fiecărui inamic în parte i-am împărțit poza corespunzătoare lui. Această metodă mi s-a părut mie simplă. Dacă aș fi pus 8 PictureBox-uri în Fereastra „Joc” era totul prea încărcat, așa că am ales să fac asta direct în cod.

Image inamic = Image.FromFile("asserts\\inamic.png");

Image inamic2 = Image.FromFile("asserts\\inamic2.png");

Image inamic3 = Image.FromFile("asserts\\inamic3.png");

Image inamic4 = Image.FromFile("asserts\\inamic4.png");

Image inamic5 = Image.FromFile("asserts\\inamic5.png");

Image inamic6 = Image.FromFile("asserts\\inamic6.png");

Image inamic7 = Image.FromFile("asserts\\inamic7.png");

Image inamic8 = Image.FromFile("asserts\\inamic8.png");

inamici[0].Image = inamic;

inamici[1].Image = inamic6;

inamici[2].Image = inamic7;

inamici[3].Image = inamic8;

inamici[4].Image = inamic5;

inamici[5].Image = inamic4;

inamici[6].Image = inamic3;

inamici[7].Image = inamic2;

* PROIECTILELE INAMICILOR

Inamicii aruncă în protagonist cu o bile de foc. Numărul nivelului la care este jucătorul este egal cu numărul de inamici care trag. Inamicii trag aleatoriu, ca în poza de mai jos:



Viteza inamicilor și viteza bilelor de foc cresc odată cu creșterea nivelului pentru a crește implicit și nivelul de dficiultate. Scorul maxim este de 200 de puncte, adică nivel 8. Când jucătorul ajunge la scorul de 200 de puncte apare un GroupBox (groupbox2) cu textul „GAME WON” și butonul de ieșire pentru a ajunge în meniu. Dacă jucătorul pierde apare același GroupBox, doar că are textul „GAME OVER”.

În ambele cazuri, când jucătorul apasă pe butonul de ieșire, dacă scorul actual făcut de jucător este mai mare decât ultimul scor reținut în baza de date, atunci scorul făcut acum va fi reținut în baza de date drept cel mai mare scor.

* COLIZIUNE

Coliziunea am întâlnit-o în trei locuri: coliziunea dintre raza Aurorei și inamicii, cea dintre Aurora și inamici și cea dintre bilele de foc ale inamicilor și Aurora. Ca să îmi fie mai usor am făcut o funcție numită „Coliziune”:

private void Coliziune()

{

for(int i = 0; i < inamici.Length; i++)

{

if (gloante[0].Bounds.IntersectsWith(inamici[i].Bounds))

{

gloante[0].Visible = false;

muzicaexplozie.controls.play();

scor++;

scorelbl.Text = scor.ToString();

levelbl.Text = "0" + level.ToString();

if (scor % 25 == 0)

{

level++;

levelbl.Text = "0" + level.ToString();

if (viteza\_inamic<=8 && viteza\_glont\_inamici<=8 && hard>=0)

{

hard--;

viteza\_inamic++;

viteza\_glont\_inamici++;

}

if (level == 8)

{

GameWon("");

}

}

inamici[i].Location = new Point((i + 1) \* 92, -100);

}

if (Jucator.Bounds.IntersectsWith(inamici[i].Bounds))

{

inimi++;

muzicainima.controls.play();

inamici[i].Location = new Point((i + 1) \* 92, -100);

muzicaexplozie.settings.volume = 5;

GameOver("");

}

}

}

Primul if verifică dacă raza Aurorei se intersectează cu un inamic, iar dacă da atunci scorul crește și evident că se modifică afișajul. Dacă scorul este divizibil cu 25 atunci nivelul crește și se schimbă afișajul și de la nivel.

Cum am spus și mai sus, odată cu creșterea nivelului crește și nivelul de dificultate, viteza inamiciilor și viteza bilelor de foc pe care le aruncă inamicii cresc. Dacă jucătorul ajunge la nivelul 8 atunci jocul este câștigat.

Dacă jucătorul se intersectează cu inamicii atunci Aurora își pierde o inimă și asta se va vedea și în joc (dispare o inimă din colțul din dreapta sus). În acest if se apelează de asemenea și funcția „GameOver”, funcție care va orpi mișcările și sunetele afișând un GroupBox cu un buton pentru a reveni în meniu și în același timp în urma apăsări butonului scorul se salvează în baza de date, doar dacă este mai mare decât cel precedent din baza de date.

private void GameOver(String str)

{

Viata();

if (inimi == 3)

{

muzicafundal.controls.stop();

StopTimers();

aipierdut = 1;

groupBox2.Visible = true;

button4.Visible = true;

Jucator.Visible = false;

}

}

private void Viata()

{

if (inimi == 1)

{

pictureBox3.Visible = false;

}

if (inimi == 2)

{

pictureBox2.Visible = false;

}

if (inimi == 3)

{

pictureBox1.Visible = false;

}

}

private void StopTimers()

{

muzicaaipierdut = new WindowsMediaPlayer();

muzicaaipierdut.URL = "songs\\sunetaipierdut";

muzicaaipierdut.settings.volume = 1;

muzicaaipierdut.controls.play();

Miscare\_background.Stop();

Miscare\_inamici.Stop();

Miscare\_glont.Stop();

Miscare\_glont\_inamici.Stop();

}

Ultima coliziune este coliziunea dintre bilele de foc lansate de inamici și Aurora. În momentul intersectării bilelor de foc cu Aurora, Aurora își mai pierde o viață și se apelează funcția „GameOver”. Am folosit o funcție pentru asta și se numește „Coliziune\_cu\_gloantele”. Această funcție am apelat-o în „Miscare\_glont\_inamici\_Tick” pentru a verifica dacă se intersectează cu Aurora.

private void Coliziune\_cu\_gloantele()

{

for (int i = 0; i < gloante\_inamici.Length; i++)

{

if (gloante\_inamici[i].Bounds.IntersectsWith(Jucator.Bounds))

{

inimi++;

muzicainima.controls.play();

int x = rnd.Next(0, 8);

gloante\_inamici[i].Location = new Point(inamici[x].Location.X + 10, inamici[x].Location.Y + 30);

muzicaexplozie.settings.volume = 5;

gloante\_inamici[i].Visible = false;

GameOver("");

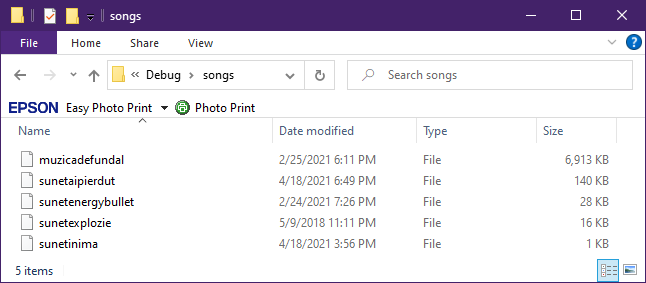
}

}

}

* SUNETE

Pe parcursul jocului am avut nevoie de muzică de fundal, sunete de efect în funcție de evenimentele care aveau loc în joc, iar aici m-am folosit de biblioteca WMPLib. Am observat că am erori dacă folosesc fișiere cu extensie mp3 pentru muzică, așa că am folosit fișiere cu muzică de tip „File”:



Voi da un exemplu de declarare și implementare al unui astfel de fișier:

WindowsMediaPlayer muzicafundal;

muzicafundal = new WindowsMediaPlayer();

muzicafundal.URL = "songs\\muzicadefundal";

muzicafundal.settings.setMode("loop", true);

muzicafundal.settings.volume = 6;

muzicafundal.controls.play();

În cazul muzicii de fundal, scopul acesteia este să ruleze pe parcursul jocului. Cum muzica mea de fundal are aproximativ 3-4 minute, am setat-o să se repete la nesfârșit și să înceapă din momentul rulării ferestrei „Joc”.

Fiecare muzică/sunet în parte are rolul lui și am considerat să folosesc și sunete deoarece face jocul mai interactiv, stârnește mai multă emoție și provoacă jucătorul să fie atent în mai multe direcții în același timp.

* PAUZĂ

În urma apăsării tastei „Esc”, apare un GrouBox (groupbox1) cu butonul de ieșire în caz că utilizatorul vrea să iasă din joc. Dacă nu, acesta poate lua o pauză deoarece inamicii nu se mai mișcă, nici nu trag, iar controalărele sunt oprite (space pentru a trage și stânga – dreapta pentru a se mișca Aurora).



La fel ca la butonul de ieșire din GroupBox-ul de joc câștigat și cel de joc pierdut, scorul se salvează dacă este mai mare decât ultimul reținu în baza de date.

**6. BIBLIOGRAFIE**

Manual: Programarea Orientată pe Obiecte și Programarea Vizuală - Microsoft