

Liceul de Informatică
“TIBERIU POPOVICIU”

Batventure

Profesori coordonatori:
Racolța Andreea
Coman Mariana

Frătean Ștefania
Clasa a XII-a A

Cuprins:

- ◆ Introducere
- ◆ Programarea Orientată – Obiect
- ◆ Interfața grafică în Unity
- ◆ Baza de date
- ◆ API
- ◆ Flow
- ◆ Ideea jocului
- ◆ Posibile îmbunătățiri
- ◆ Resurse hardware și software necesare
- ◆ Programe folosite în realizarea jocului
- ◆ Bibliografie

Introducere

Calculatorul este un instrument ce a schimbat cursul vieții a miliarde de oameni. Astăzi majoritatea populației folosește calculatorul zilnic, atât în cadrul job-ului cât și ca mijloc de recreere.

Mijloacele de recreere pe calculator implică de foarte multe ori jocuri, care sunt de mai multe tipuri precum:

- **multi-player** sau **single-player**;
- **RPG** – Role Playing Games – în acest gen fiecare jucător are un rol stabilit;
- **RTS** – adică Real-time Strategy – toți jucătorii joacă deodată, fără a-și aștepta rândul ;
- **TPS** – Turn-based Strategy – un gen caracterizat prin împărțirea în faze, fiecare jucător sau oponent AI având un timp clar în care își poate executa strategia;
- **FPS** – First Person Shooter – un gen caracterizat de vederea acțiunii prin ochii personajului;
- Arcade** – nivele simple, intuitive, cu creștere rapidă de dificultate.

Mediul de programare folosit este C#. C# derivă din două dintre cele mai de succes limbaje de programare cunoscute: Java și C++. De asemenea, este o rudă îndepărtată a limbajului C. Se poate compila prin compilatorul linie de comandă csc.exe sau utilizând mediul Visual Studio .NET (metodă utilizată și pentru aplicația prezentată). Caracteristicile limbajului C# sunt:

- Limbaj creat pentru arhitectura .NET.
- Limbaj bazat pe programarea orientată-obiect (POO)
- Limbaj simplu
- Limbaj performant
- Limbaj dinamic
- Limbaj sigur

Programare Orientată – Obiect (POO)

”Totul este un obiect.”

Limbajul C# se încadrează în categoria de Programare orientată pe obiecte (POO). Programul este un sistem de obiecte care comunică între ele prin mesaje. Fiecare obiect se caracterizează prin stare și comportament.

Fiecare obiect este o structură de date asociată cu un set de metode (atribute și metode). În programarea procedurală metodele sunt funcții sau proceduri. Invocarea (call-uirea sau apelarea) unei metode este considerată în POO ca transmiterea unui mesaj prin care i se indică operația ce trebuie făcută, transmițând de asemenea și parametrii necesari. Datele (atributele) unui obiect reprezintă starea acestuia, iar metodele indică comportamentul.

Obiectele cu aceeași structură, atribute și metode se grupează în clase. Clasa este o extindere a conceptului ”tip de dată”.

Principiile de bază ale POO sunt:

- abstractizarea: procesul de mapare a lumii reale în proiect;
- încapsularea: gruparea și ascunderea datelor care nu sunt necesare în exterior;
- moștenirea: preluarea atributelor și metodelor unei clase de către altă clasă;
- polimorfismul: capacitatea unui tip de a se comporta ca un alt tip.

Interfața grafică în Unity

Unity este o multiplatformă dezvoltată de Unity Technologies, lansată pentru prima dată în 2005.

Unity oferă posibilitatea de a crea atât jocuri 2D cât și 3D, motorul oferind un API (Application Programming Interface) primar de scripturi în C#. În cadrul jocurilor 2D, Unity permite importul de sprites și un renderer avansat 2D.

În principiu, crearea unei aplicații grafice presupune următoarele etape:

◆ Design

- Crearea unei suprafețe de afișare unde vor fi așezate obiectele grafice care servesc la comunicarea cu utilizatorul (butoane, controale, mesaje, personaje, obstacole);
- Crearea și așezarea obiectelor pe pozițiile și scenele corespunzătoare;

◆ Funcționalitate

- Definirea unor acțiuni ce trebuie executate în momentul în care utilizatorul interacționează cu obiectele grafice ale jocului;

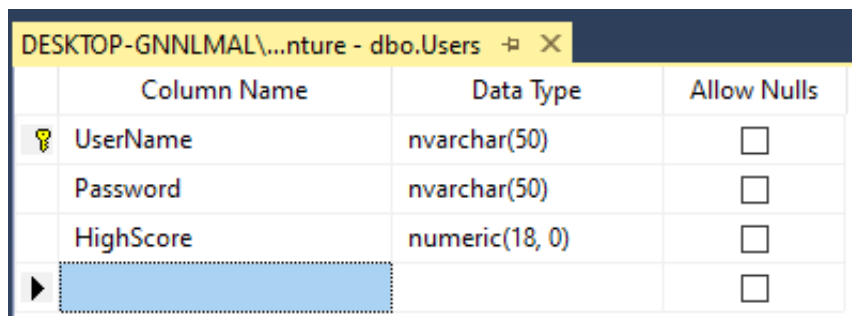
◆ Colaborarea Designului cu Funcționalitatea

- În Visual Studio, folosit pentru Unity, există anumite metode care au nume specific și sunt executate într-un anumit moment al rulării aplicației:

- ✓ void Start () – Instrucțiunile sunt executate când programul pornește;
- ✓ void Awake() – Instrucțiunile sunt executate imediat după ce primul GameObject (obiect al jocului: personaj, obstacol, element de decor, buton) este enabled;
- ✓ void Update() – Instrucțiunile sunt executate în fiecare Frame(jocul este format dintr-o succesiune de frame-uri, imagini derulate rapid astfel încât să dea impresia de continuitate).

Baza de date

Baza de date pentru acest joc constă într-un tabel cu 3 coloane: Username, Password și Highscore, care păstrează pentru fiecare user numele de utilizator, parola și cel mai mare scor obținut



DESKTOP-GNNLMAL\...nture - dbo.Users			
	Column Name	Data Type	Allow Nulls
🔑	UserName	nvarchar(50)	<input type="checkbox"/>
	Password	nvarchar(50)	<input type="checkbox"/>
	HighScore	numeric(18, 0)	<input type="checkbox"/>
▶			<input type="checkbox"/>

până în prezent, având ca primary key Username-ul. Primary key-ul este identificatorul unic pentru fiecare rând din tabel.

Utilizatorul are acces la 3 operații pe baza de date: crearea unei linii noi, interogarea bazei de date și un update pe o linie.

Crearea unui user înseamnă crearea unui cont nou și constă în adăugarea unei linii noi în baza de date care să dețină username-ul dorit, parola și inițializează automat highscore-ul cu 0. Pentru ca acest cont să fie creat trebuie îndeplinită următoarea condiție la introducerea datelor: numele de utilizator trebuie să fie unic în baza de date. În cazul în care se introduce un Username deja existent crearea va eșua.

Pentru crearea unei noi linii în baza de date se folosește metoda INSERT. În cazul creării unui User cu Username-ul "user1" și parola "password1" se execută următorul query:

***INSERT INTO Users (UserName, Password, Highscore)
VALUES ("user1", "password1", "0")***

În momentul logării trebuie verificat dacă user-ul există și dacă parola introdusă corespunde celei din baza de date. Pentru aceasta se folosește operația SELECT după Username, care este și primary key. Pentru logarea cu Username-ul "user1" și parola "password1" se execută următorul query:

***SELECT *
FROM Users
WHERE UserName = "user1"***

Validarea existenței username-ului și a parolei se va face în API.

Cel de-al treilea tip de query este un update pe o linie deja existentă în DB. Update-ul se face pe coloana Highscore, iar query-ul folosit pentru *user1* cu highscore 10 este următorul:

***UPDATE Users
SET Highscore = 10
WHERE UserName = "user1"***

Ultimul Query folosit este cel ce returnează datele pentru Scoreboard. Acest query returnează toți userii, ordonarea urmând a se face în API. Query-ul folosit este următorul:

***SELECT *
FROM Users***

Folosind C# există un Framework de abstractizare a bazei de date, denumit Entity Framework. Acesta va crea modelul după baza de date, care va fi folosit în API.

API

Termenul API este un acronim și reprezintă Application Programming Interface (Interfața de Programare a Aplicațiilor). Un API enumeră un set de operații pe care dezvoltatorii le pot utiliza, împreună cu o descriere a ceea ce fac. Dezvoltatorul nu trebuie neapărat să știe cum funcționează în spate, ci doar că o are la dispoziție.

API-urile sunt, de asemenea, folosite pentru a abstractiza implementarea și pentru a controla accesul la dispozitivele hardware și funcțiile software pe care o aplicație nu are neapărat permisiunea de a le utiliza. De aceea, API joacă adesea un rol important în securitate. Astfel, API permite ca doar aplicațiile dorite să fie făcute publice.

API este un proiect care rulează pe un server HTTP (HyperText Transfer Protocol). HTTP este un protocol utilizat de WWW (World Wide Web) care definește cum sunt formate și transmise mesajele și ce acțiuni trebuie să execute serverele Web și browser-ele pentru a răspunde la diferite comenzi. Serverul pe care rulează API-ul are o adresă și un port. Având DB-ul local, adresa lui este un localhost. Având baza de date locală, are sens ca și serverul să fie tot local, având astfel o adresă localhost, iar portul este 49184.

API-ul a fost introdus pentru a nu conecta direct jocul la baza de date.

Comunicarea între joc și API se face prin call-uri HTTP la endpoint-uri. Un endpoint API este punctul de intrare în canalul de comunicare când cele două sisteme interacționează. Astfel, metodele HTTP pot fi observate înaintea query-urilor, fiind de mai multe tipuri, mai exact:

- GET – primește date de la server, pentru o resursă specificată
- POST – creează o nouă entitate
- PUT – updatează o dată deja existentă
- DELETE – șterge date
- PATCH – face un update parțial pe o dată

În aplicația prezentată am folosit doar două dintre aceste tipuri, mai exact GET (pentru operațiile folosite în timpul logării și afișării scoreboard-ului) și POST pentru crearea unui nou cont și update-ul Highscore-ului.

Serializarea request- response între joc și API se face într-un format numit JSON. Sintaxa JSON este de fapt un set de sintaxe JavaScript, funcționând pe principiul "nume-valoare". O pereche name/value constă într-un câmp de nume (unde e numele valorii) între ghilimele, urmat de semnul ":", urmat de valoare, de asemenea între ghilimele. Un JSON constă într-un anumit număr de astfel de

perechi, așezate succesiv între acolade, despărțite prin virgulă. Un exemplu de JSON este:

```
user = {"Username": "user", "Password": "1234", "Highscore": "12"};
```

Flow

- Jucătorul îți crează un cont nou, care va fi adăugat în baza de date. Acesta trebuie să introducă username-ul, parola și apoi să o confirme (acestea trebuie să fie identice).

Jocul va face un request de tip POST către API cu parametrii Username și Password. API-ul va crea în baza de date un nou user cu datele introduse de utilizator și highscore-ul 0. Crearea unui user se poate executa cu succes, iar atunci API-ul va returna User-ul nou, sau poate arunca o excepție, returnându-se în acest caz *null*.

```
[HttpPost]
public User CreateUser(string Username, string Password)
{
    using (batventureEntities model = new batventureEntities())
    {
        User newUser = new User();
        newUser.HighScore = 0;
        newUser.Password = Password;
        newUser.UserName = Username;

        model.Users.Add(newUser);
        try
        {
            model.SaveChanges();
        } catch (Exception)
        {
            return null;
        }
        return newUser;
    }
}
```

- Dacă jucătorul are deja un cont, se poate loga cu username și parolă. Jocul va face un request de tip GET către API cu parametrii Username și Password. API-ul caută în baza de date Username-ul (care este unic) și verifică dacă parola asociată acestuia este identică celei introduse de utilizator. Dacă Parolele sunt aceleași, este returnat User-ul și se execută logarea. În cazul în care nu corespund, se returnează o valoare nulă


```

13 public IEnumerable<User> getAll()
14 {
15     using (batventureEntities model = new batventureEntities())
16     {
17         return model.Users.ToList();
18     }
19 }
20
21 public User GetUserAndPassword(string Username, string Password)
22 {
23     using (batventureEntities model = new batventureEntities())
24     {
25         User userInstance = model.Users.Find(Username);
26         if(userInstance != null)
27         {
28             if(userInstance.Password == Password)
29             {
30                 return userInstance;
31             }
32         }
33         return null;
34     }
35 }

```

- La finalul unui joc, după ce liliacul lovește un obstacol se calculează scorul. Jocul face un request la API de tipul POST cu parametrii Username și Highscore, se caută user-ul după UserName și se verifică dacă scorul nou obținut este mai mare decât highscore-ul obținut anterior. Dacă noul scor este mai mare se modifică în baza de date câmpul corespunzător highscore-ului. Indiferent dacă modificarea a avut loc sau nu la final se returnează user-ul.

```

59 [HttpPost]
60 [Route("api/user/highscore")]
61 public User UpdateHighscore(string Username, int Highscore)
62 {
63     using(batventureEntities model = new batventureEntities())
64     {
65         User userInstance = model.Users.Find(Username);
66         if(userInstance.HighScore < Highscore)
67         {
68             userInstance.HighScore = Highscore;
69         }
70         model.SaveChanges();
71         return userInstance;
72     }
73 }

```

- La final, pentru afișarea clasamentului, jocul face un request către API de tip GET, care returnează toți userii în ordine descrescătoare a highscore-ului, în cazul în care sunt mai puțini decât 10, sau primii 10 useri cu cel mai mare scor în cazul în care numărul lor este mai mare decât 10.

```

[HttpGet]
[Route("api/user/scoreboard")]
public IEnumerable<User> GetScoreboard()
{
    using (batventureEntities model = new batventureEntities())
    {
        IEnumerable<User> userList = model.Users.ToList();
        IEnumerable<User> orderedUsers = userList.OrderByDescending(u => u.HighScore).ToList().GetRange(0, userList.Count() >= 10 ? 10 : userList.Count());
        return orderedUsers;
    }
}

```

Ideea Jocului

Batventure este un joc Arcade cu idee simplă. Utilizatorul controlează un liliac care se deplasează de la stânga la dreapta, zburând printre obstacole cu ajutorul Space-ului, buton care face liliacul să se înalțe. Obiectivul jocului este trecerea printre stalactite și stalagmite, fără a atinge vreuna. Scorul crește de fiecare dată când liliacul trece de un obstacol. În cazul atingerii, jocul se încheie, jucătorul fiind nevoit să o ia de la capăt, cu scorul 0. Jocul se continuă la infinit, repetându-se primele 13 stalactite.

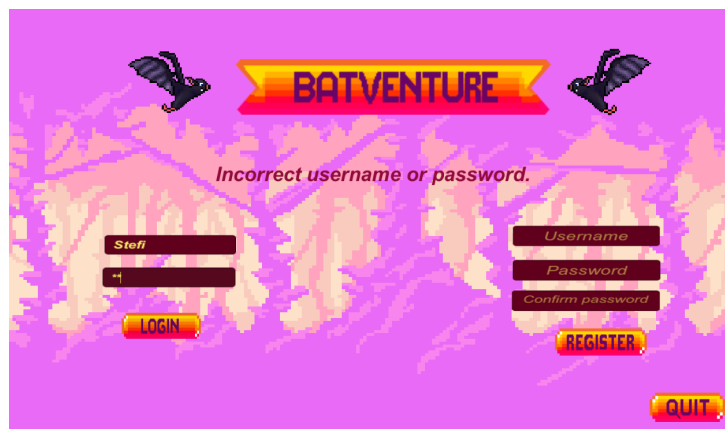
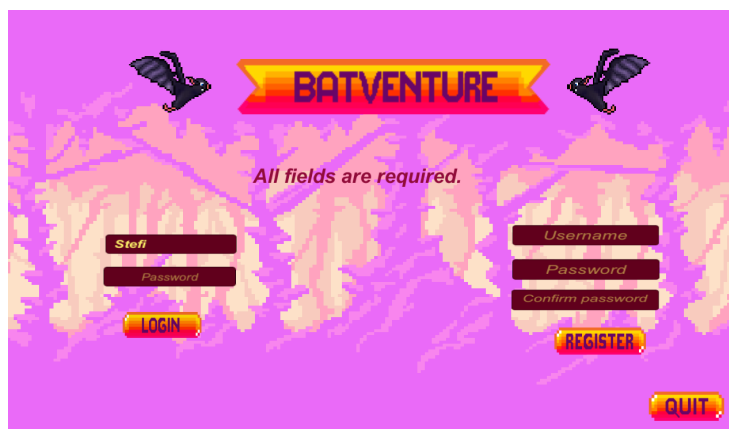
Dificultatea crește de fiecare dată când jucătorul mai câștigă 10 puncte (adică la scorul 10, 20, 30, etc) prin mărirea vitezei de deplasare a liliacului.

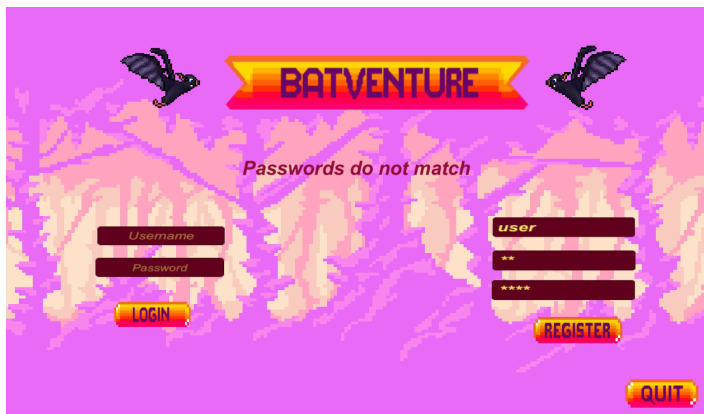
Pentru ca fiecare utilizator să își poată păstra cel mai bun scor, acesta își creează un cont la începutul jocului (folosind meniul de deasupra butonului **”Register”** și apăsându-l ulterior) sau se conectează dacă are unul deja creat (folosind meniul de deasupra butonului **”Login”** și apăsându-l ulterior).



În cazul înregistrării se adaugă un nou rând în baza de date, iar în cazul conectării se verifică corectitudinea datelor introduse cu cele din tabel.

Se afișează diferite mesaje pentru a ajuta userul să introducă date corecte.





Odată conectat, jucătorului îi apare scena de început de joc, unde are două butoane: Start și Quit.

La începutul jocului, sunt prezentate instrucțiunile și utilitatea tastelor:

- **Space** – pentru a juca, fiind butonul care face liliacul să se înalțe în timpul jocului;
- **Esc** – pentru a opri temporar jocul. Dacă este apăsat o data apare meniul de pauză, cu alte 3 butoane, iar dacă este apăsat a doua oară, începe instant jocul de unde a rămas. Meniul de pauză conține următoarele butoane:

- **"Resume"** – reia jocul de unde a rămas;
- **"Menu"** – face jocul să se întoarcă la pagina de start, fără a salva însă scorul obținut până în acel moment. Mai exact anulează jocul în curs;
- **"Quit"** – duce jucătorul pe pagina de final, unde este afișat scorul și tabelul cu primii 10 jucători care au cel mai mare scor.

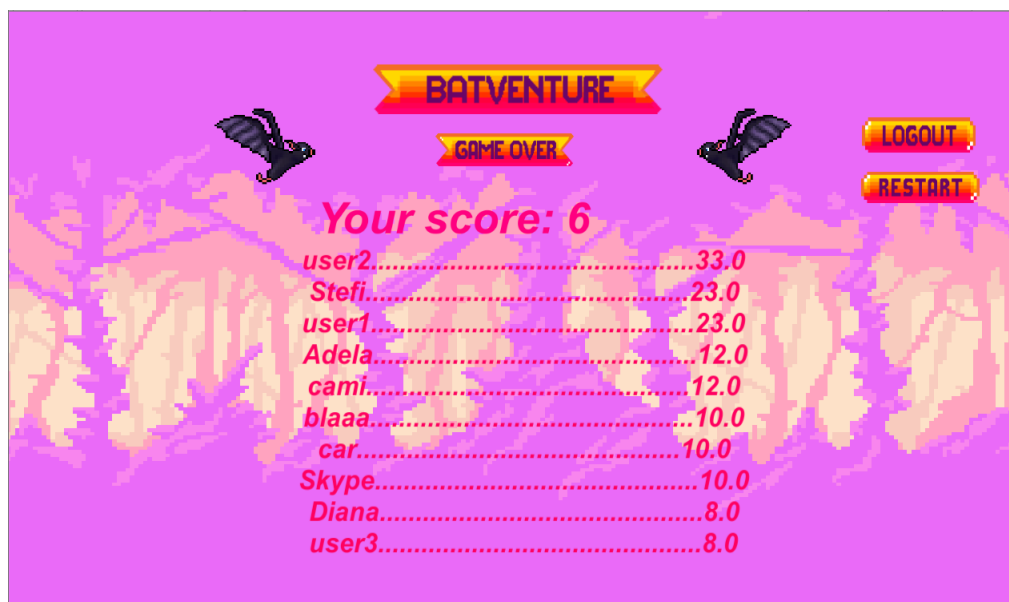


În colțul din dreapta sus, utilizatorul își poate vedea scorul pe tot parcursul jocului.

De asemenea, pe scena de final se afișează scorul obținut în jocul tocmai terminat și tabelul cu primii 10 jucători care au obținut cele mai mari scoruri. Tot pe

această scenă există și butonul **"Restart"** care permite reînceperea jocului, astfel userului permițându-i-se să joace ori de câte ori dorește.

Pentru a ieși din aplicație, utilizatorul trebuie să încheie un joc pentru a ajunge pe pagina de final și să apese butonul **"Logout"**. Acesta îl duce pe pagina de logare, unde există butonul **"Quit"**. Odată apăsă, aplicația se închide.



Posibile îmbunătățiri

Un posibil upgrade care poate face jocul mai atractiv ar fi schimbarea ordinii în care apar stalactitele pe ecran. Momentan, 13 stalactite se repetă la infinit în aceeași ordine. Acestea ar putea apărea însă random la o distanță oarecare între ele, pentru a nu lăsa jucătorul să se acomodeze cu traseul, cerând mult mai multă atenție pentru a nu lovi obstacolele chiar dacă utilizatorul a mai încercat jocul înainte.

O altă îmbunătățire posibilă ar fi adăugarea unei coloane sonore. O melodie de fundal, sunete la apăsarea butoanelor, sunete în momentul în care liliacul atinge un obstacol.

De asemenea, ar putea fi făcut funcțional butonul tab în momentul logării, pentru a nu mai fi nevoie de mouse pentru a trece de pe un field pe altul.

Resurse hardware și software necesare

- Server:
 - 1GB spațiu de stocare
 - 2GB RAM
 - procesor de minimum 1.66 GHz;
- Joc:
 - procesor de minimum 1.66 GHz;
 - SQL Server 2019 Configuration Manager
 - 635 KB spațiu de stocare
- Conexiune la internet

Programe folosite în realizarea jocului

- Adobe Photoshop CC 2015
- Unity 2018.3.7f1
- Visual Studio 2019
- SQL Server 2019 Configuration Manager
- Microsoft SQL Server Management Studio

Bibliografie

- <https://stackoverflow.com/questions>
- <https://assetstore.unity.com/?category=2d&orderBy=1>
- <https://www.youtube.com/watch?v=YQ7Umjp6R10>
- https://www.youtube.com/watch?v=_brBV61f2ss&t
- <https://www.youtube.com/watch?v=xvLMD2qWaKk&ts>
- <https://www.youtube.com/watch?v=on9nwbZngyw>
- <https://www.youtube.com/watch?v=CJ8FKjYtrT4>