

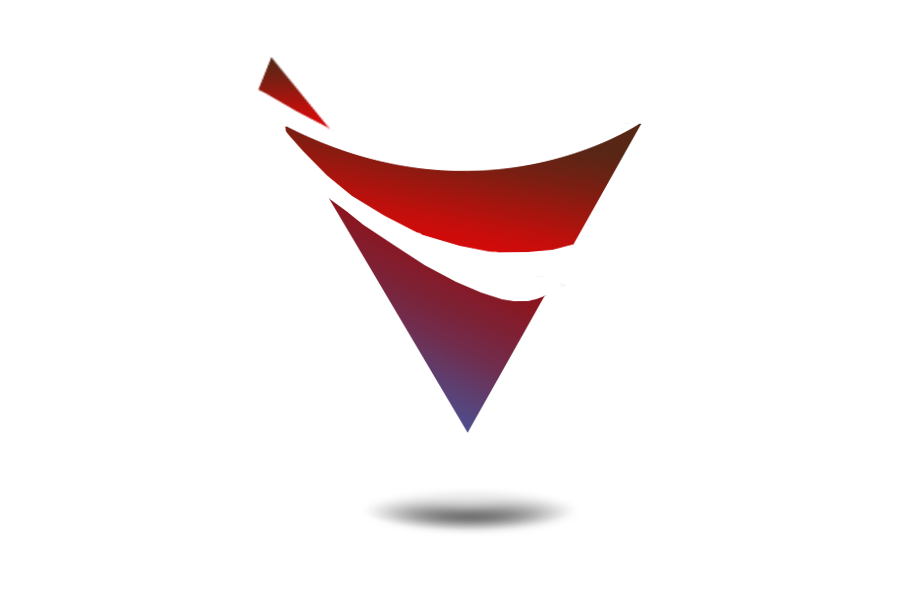
LUCRARE DE ATESTAT DISCIPLINA INFORMATICĂ

Candidat Coordonator

Voia Andrei Prof. Gal Carmen

2016

APLICAŢIA CHOWDER



Cuprins:

[1.INTRODUCERE 4](#_Toc450718757)

[2.PREZENTAREA MEDIULUI DE PROGRAMARE 6](#_Toc450718758)

[3. DESCRIEREA PĂRȚII PRACTICE 9](#_Toc450718759)

[3.1. Utilizarea Aplicației 11](#_Toc450718760)

[3.2. SUBPROGRAME / ALGORITMI UTILIZAȚI 14](#_Toc450718761)

[4. CONCLUZII 28](#_Toc450718762)

[5. BIBLIOGRAFIE 30](#_Toc450718763)

# 1.INTRODUCERE

Am ales să creez un joc, ulterior denumit CHOWDER, deoarece sunt pasionat de jocurile video și dintotdeauna mi-am dorit sa vad ce se află “în spatele ecranului”, care este “mecanismul” unui joc și cum se poate realiza. De asemenea, CHOWDER este desenul meu preferat pe care îl vizionam constant în copilăria mea, astfel a devenit și tema jocului pe care l-am conceput.

CHOWDER este un joc conceput pentru platforma Windows, realizat în C#, care iți pune la încercare îndemânarea și are ca scop principal divertismentul jucătorului. În acest joc te afli în posesia lui Chowder, un ucenic care își dorește să devină un mare maestru bucătar. Drumul până acolo, însă, este lung, iar pentru a învăța și pentru a acumula experiență, Chowder se află în slujba maestrului său, Mung Daal.

În jocul care-i poartă numele, CHOWDER, trebuie să aduni toate fructele pe care le găsești și să i le aduci lui Mung Daal pentru a pregăti o prajituă specială după rețeta sa secretă. Fructele se află în cutii pe care trebuie sa le spargi. Totuși, fii cu băgare de seamă, Schnitzel și Panini vor să te oprească cu orice preț, ferește-te de ei. De asemenea, în CHOWDER ți se oferă posibilitatea să te joci împreună cu prietenul tău, acesta poate să intre în pielea lui Gorgonzola, unul dintre personajele, care, este invidios pe protagonistul jocului și dorește să îl împiedice din drumul său, chiar dacă asta înseamnă să îi fure fructele de care are nevoie pentru tort, cât și să arunce după el cu pește, singura mâncare pe care Chowder nu o poate mânca.

Jocul este împărțit în 6 nivele, dificultatea crescând odată cu nivelul selectat de către jucător. La intrarea în joc este afișat pe ecran un meniu în care ți se explică butoanele folosite, cum să adaugi un “player 2” precum și ce trebuie să faci pentru a ieși câștigător. În ceea ce privește coloana sonoră a acestui joc, a fost preluată din desenul original Chowder și prelucrată pentru a se îmbina cu ritmul alert al jocului, pentru a-i da un aer dinamic.

Jocul CHOWDER a fost realizat în C# datorită posibilităților oferite de această platformă de programare în realizarea produsului finit, împachetat într-un ambalaj plăcut vizual.

Cerințe de sistem minime și recomandate pentru rularea jocului:

CPU / Procesor: Intel Pentium Dual Core 1.5 Ghz

Recomandat: AMD FX 4100 Black Edition 3.5 Ghz

CPU Speed / Viteză Procesor: 1.5 Ghz, Dual Core

Recomandat: 2.5 Ghz, Dual Core sau mai bun

RAM: 512MB

Recomandat: 1.5GB, 1600 Mhz

Placă Video: 255 VMB sau mai mult

Recomandat: AMD Sapphire Radeon HD 6790, 1GB GDDR5 / Nvidia GIGABYTE GeForce GTX 660, 2GB GDDR5

Sistem de operare: Windows XP – 64bit – Service Pack 3

Recomandat: Windows 7 - 64bit – Service Pack 1 / Windows 8 – 64bit – orice Sevice Pack / Windows 10 – 64bit – orice Service Pack

Versiune DirectX: 9.0c compatibil

Recomandat: DirectX 11 compatibil

Versiune .NET Framework: 4.0

Recomandat: .NET Framework 4.5

Visual Stidio C#: Nu / Opțional

Placă de Sunet: Da

Spațiu liber pe HDD: 30 MB

HDD: 5400 RPM / Rotații Pe Minut

Recomandat: 7200 RPM / Rotații Pe Minut

Display: rezoluție 1280 x 720, 30 Hz, 5 ms

Recomandat: rezolutie 1440 x 900, 65+ Hz, 1 ms

DVD-ROM: Da

Programe utilizate în realizarea jocului:

Visual Studio C# 2013 -64bit

Photoshop CS6 -64bit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# 2.PREZENTAREA MEDIULUI DE PROGRAMARE

C# este un limbaj de programare care combină facilități testate de-a lungul timpului cu inovații de ultim moment. Denumirea sa a fost inspirată din notația # (diez) în muzică, indicând faptul că este mai înalt decât C++. Lănsat în 2000 și oficial în primăvara anului 2002, C# a fost dezvoltat de către Microsoft, respectiv Anders Hejlsberg, Scott Wiltamuth și Peter Golde.

Deși limbajul este creat de Microsoft, producătorul platformei Windows, acesta nu este destinat doar platformelor Microsoft, existând compilatoare și pentru alte sisteme, precum Linux.

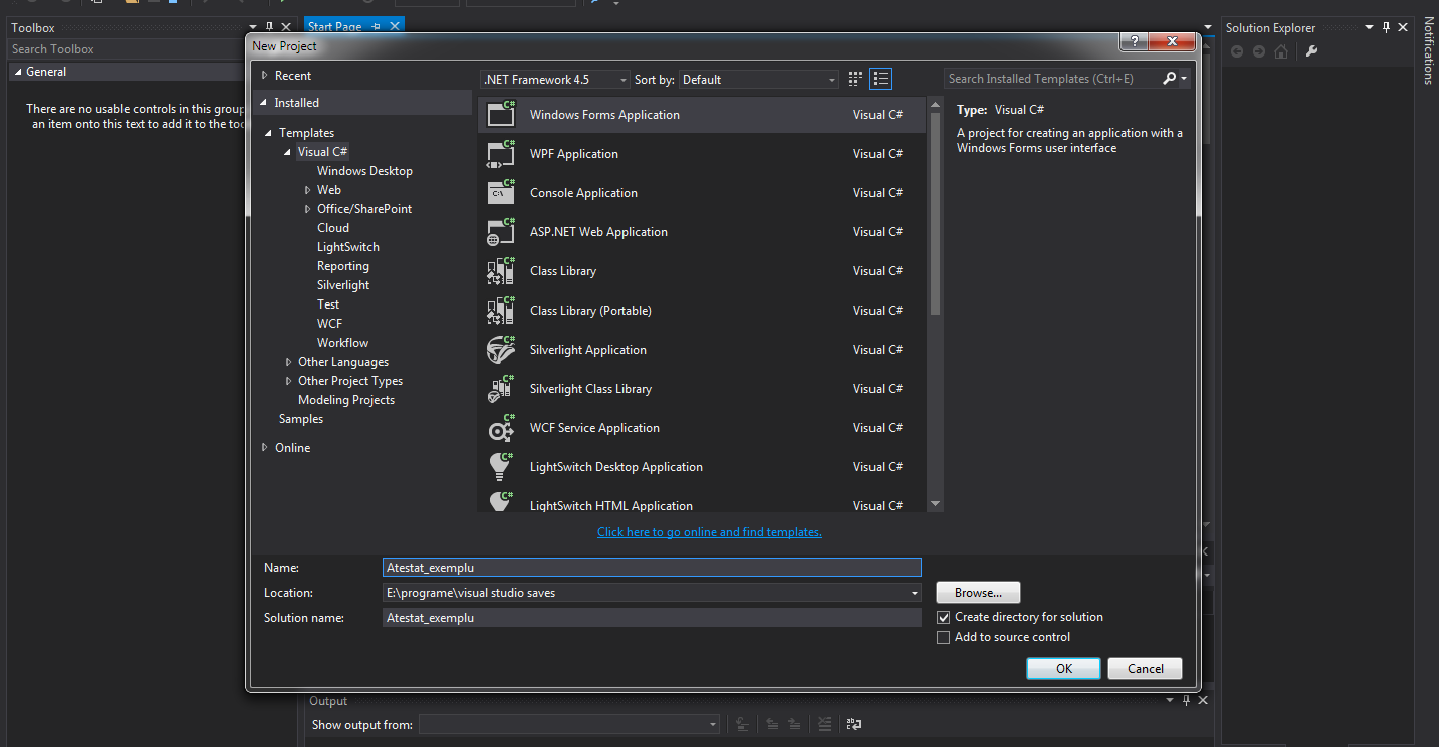
Creat ca instrument de dezvoltare pentru arhitectura .NET Framework, limbajul oferă o modalitate facilă și eficientă de a scrie programe pentru sistemul Windows, internet, componente software și altele.

C# gestionează în mod automat memoria utilizată. Eliberarea memoriei ocupate, denumită „garbage collection”, de către obiectele care nu mai sunt necesare aplicaţiei, este o facilitate importantă a limbajului. Programatorii nu mai trebuie să decidă singuri, aşa cum o fac de pildă în C++, care este locul şi momentul în care obiectele trebuiau sa fie distruse.

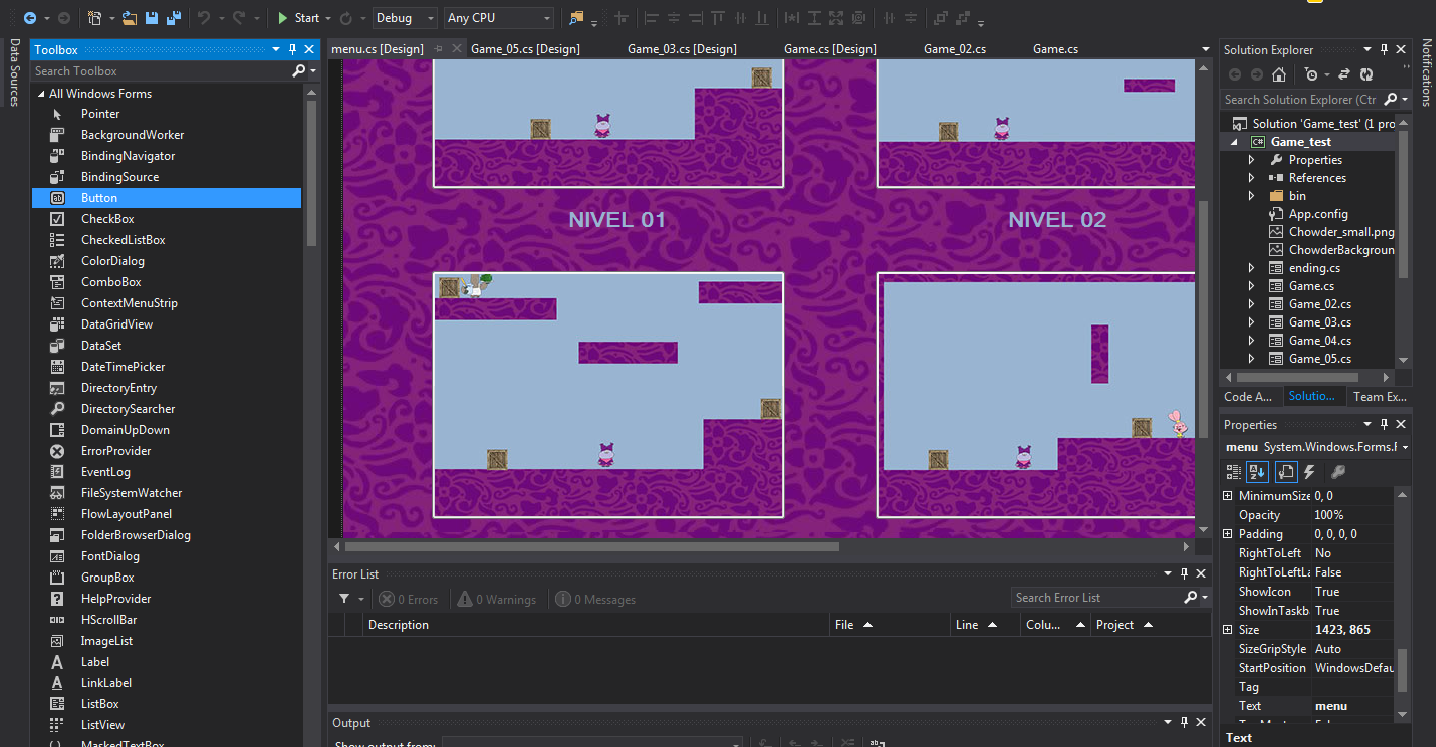
Limbajul C# este un limbaj orientat pe obiecte, sau object oriented programming, abreviat OOP. Avantajul este faptul că pot fi create noi tipuri de date. O clasă reunește coduri și date care descriu o nouă categorie de obiecte, crearea unui nou obiect numindu-se instanțiere iar obiectul creat numindu-se instanță a clasei. Mecanismele fundamentale ale OOP sunt încapsularea, polimorfismul, moștenirea și reutilizarea.

Visual C# este o aplicaţie Windows, deci foloseşte elemente de interfaţă specifice acestui mediu de lucru: ferestre, meniuri, butoane și obiecte de control. C# beneficiază de o interfață practică și ușor de folosit datorită faptului că a fost lansat cu câțiva ani în urma limbajului de programare Java, astfel echipa care a avut în grijă dezvoltarea platformei a avut timp să îl conceapă sub o formă de tipul „user friendly”.

# 3. DESCRIEREA PĂRȚII PRACTICE

În urma lansării programului, avem posibilitatea de a ne crea un nou proiect cu un singur click pe „New Project”, apoi, după ce ne decidem asupra numelui, selectăm „Windows Forms Application” și apăsăm pe butonul „OK”. Asta a fost tot, acum să dai frâu liber imaginației și să „construiești” programul pe care ți-l dorești.

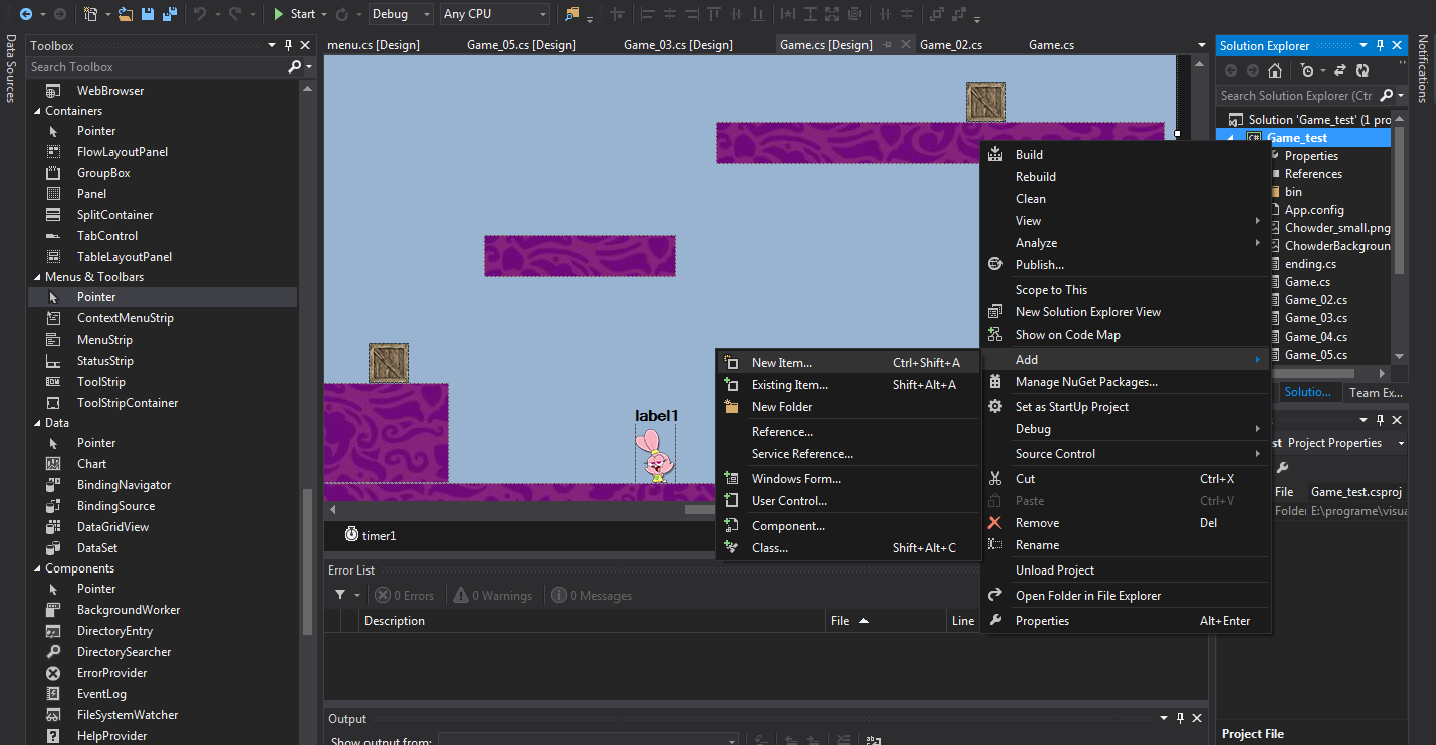
Ne sunt oferite multe informații prin intermediul meniului descriptiv de tip „popup” care apare atunci când ținem cursorul deasupra unui buton, astfel, ne sunt afișate, succint, informații referitoare la acțiunea respectivă. Avem o multitudine de opțiuni atunci când vine vorba de abordarea unei probleme iar, pentru crearea interfeței, posibilitățile sunt la fel de numeroase. În Visual C#, prin intermediul ferestrei „Toolbox”, putem adăuga elemente esențiale pentru orice aplicație creată ulterior, precum:

* Button / Buton
* Label / Etichetă
* Timer / Cronometru
* PicureBox / Imagine
* Form / Fereastră

De asemenea, trebuie să ținem cont de toate fișierele folosite pentru realizarea programului, acestea fiind gestionate în cadranul „Solution Explorer”, de obicei găsit în partea dreapta-sus a ecranului. Aici se află, printre altele, și:

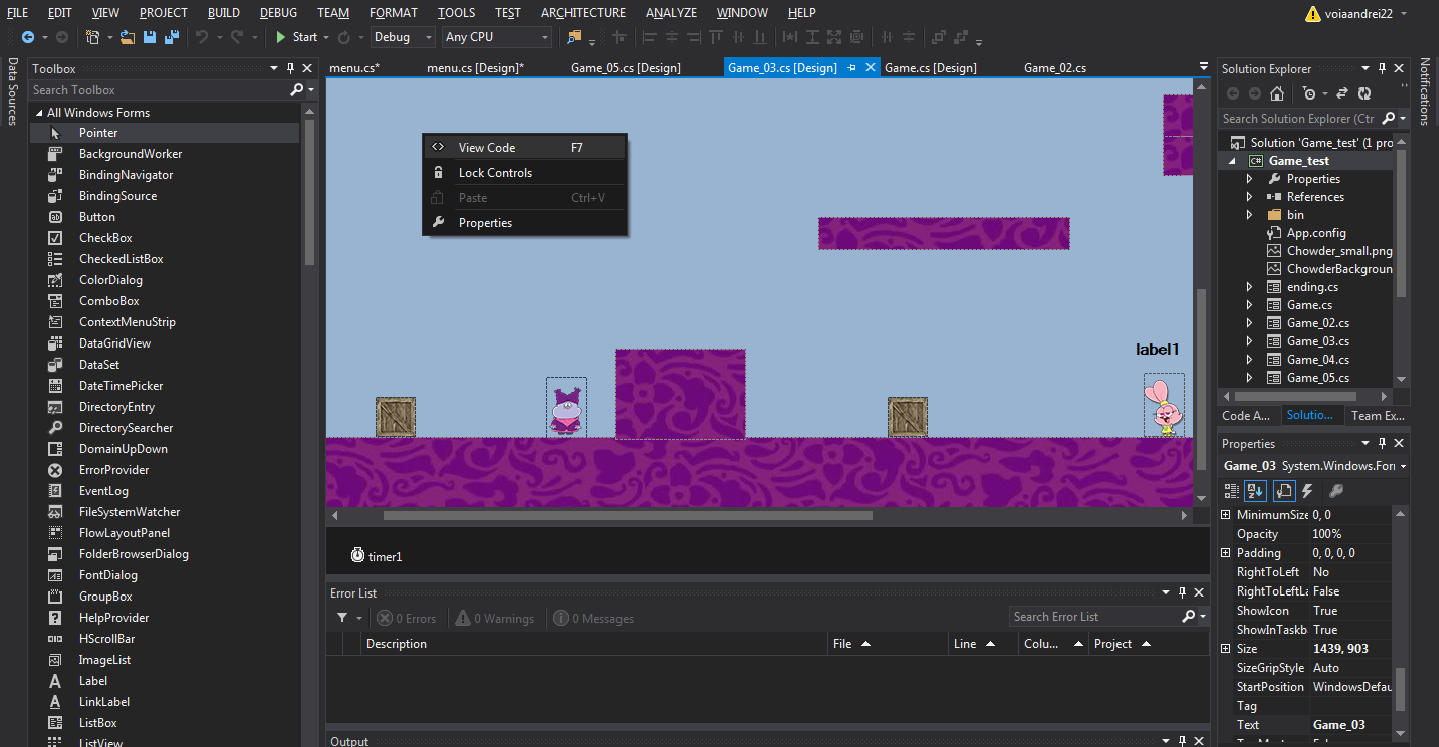
* Forms / Ferestrele
* Properties / Proprietăți
* References / Referințe
* Imaginile și celelalte fișiere, cele audio, de exemplu, folosite în realizarea jocului

Introducerea unei ferestre noi, sau a unui nivel, în cazul acestui joc, se face din meniul „Solution Explorer”, executându-se click dreapta pe numele proiectului, respectiv „Game\_test”, după care executăm click stânga pe „Add”, apoi pe „New Item…”. În acest fel putem adăuga multe ferestre secundare pe care le putem folosi așa cum dorim.

Butoanele au fost esențiale în realizarea meniului acestui joc deoarece, prin intermediul lor, s-a făcut trecerea la nivelul respectiv. Butoanele pot fi aplicate în proiect prin dublu click pe numele „Button” din „Toolbox”.

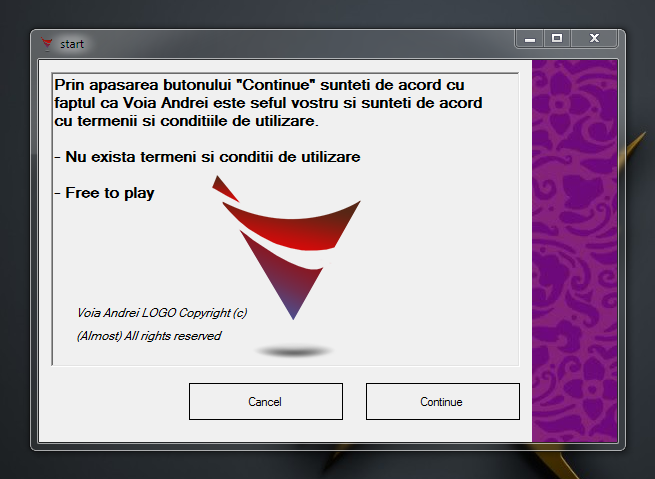
Toate imaginile folosite, atât pentru personajele din CHOWDER cât și pentru conturarea mediul înconjurător, se introduc prin „PictureBox”-urile care ne sunt puse la dispoziție de către meniul „Toolbox”.

Etichetele au reprezentat partea descriptiva a jocului, acestea gasindu-se in fiecare fereastra, de la meniul principal si pana la ultimul nivel. In meniu, etichetele au fost redenumite în funcție de nivelul pe care îl reprezintă, iar în interiorul nivelelor, ele au fost plasate astfel încât să se portivească atât cu replicile caracterelor cât și cu numele și procentul de viață al ucenicului, adică al lui Chowder.

 Nicio acțiune și nicio animatie nu ar fi fost posibilă fără folosirea unui „Timer”. Acesta se găsește, precum toate celelalte elemente, în „Toolbox” și se folosește la deplasarea de n ori într-o secundă a mai multor obiecte.

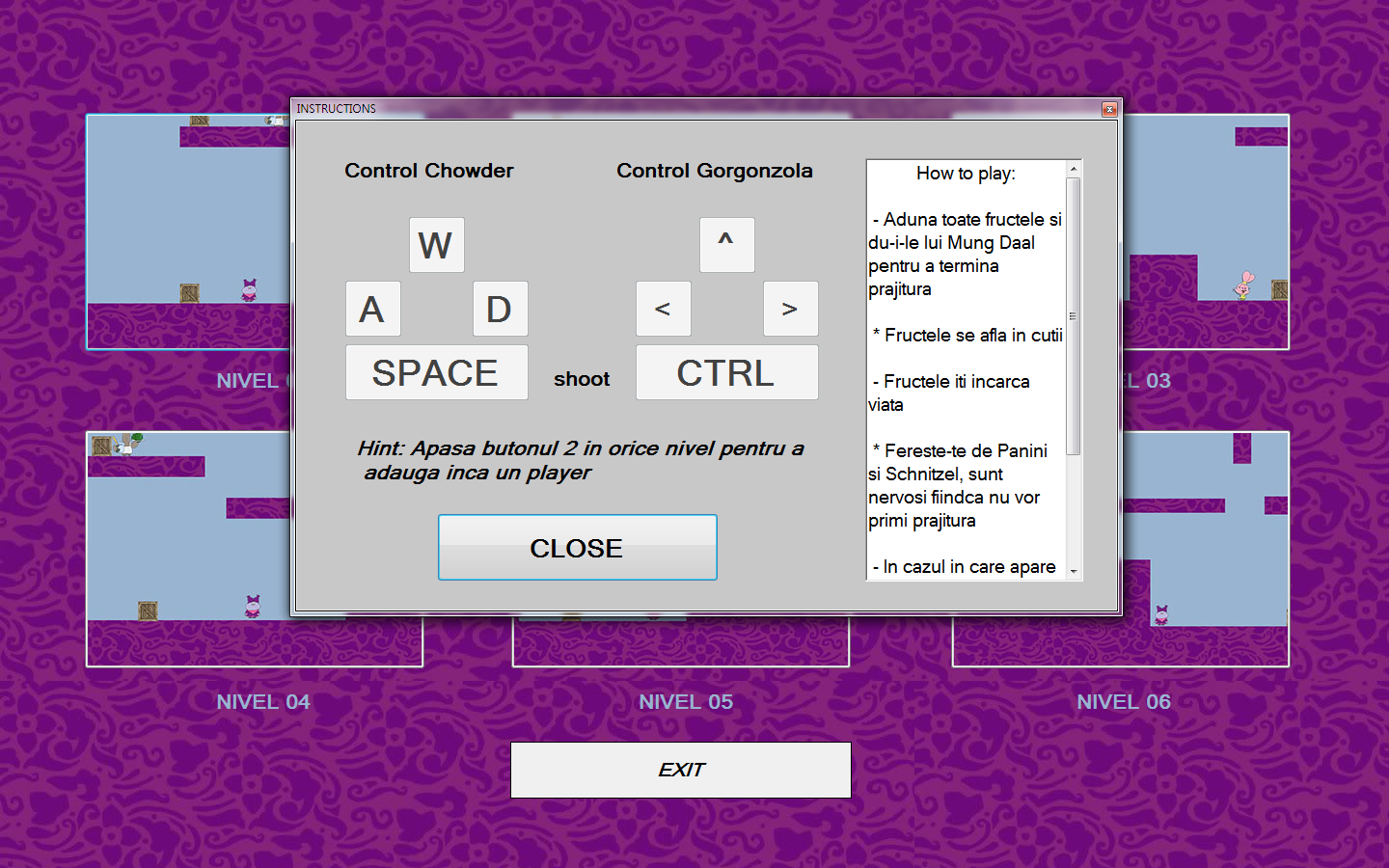
Cea mai simplă modalitate prin care se poate ajunge în interiorul programului propriu-zis este printr-un simplu click dreapta pe fereastra al cărui cod doriți să fie afișat, urmat de apăsarea butonului „View Code”. Acest lucru permite modificarea programului. De asemenea, în cazul în care programul are erori, acestea vor fi afișate, de regulă, în partea de jos a ecranului, în categoria „Error List”.

## 3.1. Utilizarea Aplicației

Pentru lansarea aplicației CHOWDER nu este necesară prezența limbajului de programare C# pe calculatorul sau pe laptopul respectiv. Există trei moduri prin care acest joc poate fi rulat:

1. Prima modalitate presupune existența programului „Visual C#” dar acest lucru are un avantaj, și anume că admite modificarea programului în cod.
2. Cea de-a două opțiune este instalarea prorpiu-zisă a jocului folosind fișierul „Setup.exe”, prezent pe CD-ul de instalare.
3. Ultima și, de altfel, metoda mea preferată, este rularea executabilului „Chowder\_Launcher.exe”, fără a fi necesară o instalare precedentă.

În urma lansării aplicației, pe ecran va apărea o fereastră mică pe care sunt prezentați termenii și condițiile de utilizare, precum și LOGO-ul meu. Jocul va porni în urma apăsării butonului „Continue” iar, în mod evident, butonul „Cancel” va ieși din program.

 CHOWDER va porni prin afișarea, în format „Fullscreen”, a meniului jocului. Acest meniu conține toate nivelele accesibile de către jucător iar acestea pot fi selectate printr-ul simplu click pe imaginea lor. Există și un buton pentru a ieși din joc, iar acesta se numește „EXIT”.

De asemenea, acoperind meniul jocului, va fi afișată o fereastră, denumită „INSTRUCTIONS”, care, precum sugerează și numele său, prezintă butoanele folosite de către jucător și informații esențiale referitoare la nivele.

Fiecare nivel este special construit pentru a-ți testa abilitățile și viteza de reacție, precum și pentru a te face să te simți bine în timp ce il parcurgi, muzica constituind un element foarte important, crucial al acestei platforme.

Fiecare nivel vine cu posibilitatea de a fi jucat în mod „competitive”, împreună cu o persoană care poate intra în lumea lui Chowder, punându-se în pielea lui Gorgonzola. În acest caz modul jocului se schimbă, devenind un joc eliminatoriu. Ultimul rămas în viața castigă iar acest lucru poate fi realizat prin eliminarea adversarului înainte ca el să te elimine pe tine.

## 3.2. SUBPROGRAME / ALGORITMI UTILIZAȚI

1. public Game()
2. #region //initializare
3. void resolution(int x, int y)
4. void if\_resolution\_differs()
5. void if\_resolution\_differs\_add\_player\_02()
6. private void Form1\_KeyDown(object sender, KeyEventArgs e)
7. void add\_player\_02()
8. private void Form1\_KeyUp(object sender, KeyEventArgs e)
9. void right\_left\_move\_player\_02()
10. void jumping\_player\_02()
11. void collision\_01\_player\_02(PictureBox block)
12. void collision\_02\_player\_02(PictureBox block)
13. void bullet\_moving\_player\_02()
14. void collect\_player\_02(PictureBox item, int count)
15. void enemy\_collision\_player\_02(PictureBox enemy)
16. void enemy\_02\_collision\_player\_02(PictureBox enemy)
17. private void timer1\_Tick(object sender, EventArgs e)
18. void right\_left\_move()
19. void bullet\_moving()
20. void jumping()
21. void collision\_01(PictureBox block)
22. void collision\_02(PictureBox block)
23. void collect(PictureBox item, int count)
24. void enemy\_move(PictureBox enemy, int lft, int rgt)
25. void enemy\_move\_02(PictureBox enemy, int lft, int rgt)
26. void MungDaal\_finish()
27. void platform\_move(PictureBox floating, int up, int down)
28. void limits(PictureBox person)

Inițial, am creat interfața vizuală a jocului, am generat, dacă a fost necesar, caracterul Gorgonzola, precum și toate label-urile folosite în nivelul respecitv. De asemenea, la fiecare ieșire dintr-un nivel, acesta se închide pentru a nu ocupa memorie RAM și pentru a nu „mânca” din capacitatea procesorului.

1. *Acțiunile*, „**public Game()**”, care au loc în momentul în care aplicația este pornită sunt:

* Inițializarea întregului text prezent pe interfața programului precum și localizarea lui
* Definirea solului, adică nivelul înălțimii de la care nu se poate trece în jos
* Modificarea rezoluției în funcție de numărul de pixeli de care dispune dispozitivul extern de ieșire, respectiv monitorul

……………………………….

Cursor.Hide();

life\_player\_02\_text.Left = -200;

inventar\_player\_02.Left = -200;

Gorgonzola\_name.Left = -200;

chowder\_player.Hide();

chowder\_player.SendToBack();

chowder\_player.Left = -200;

gorgonzola\_player.Hide();

gorgonzola\_player.SendToBack();

gorgonzola\_player.Left = -200;

……………………………….

1. Toate *inițializările* au fost introduse într-un **„#region – #endregion**” pentru a le deosebi de restul subprogramelor cu ușurință, acestea fiind pentru:

* Locația personajelor
* Locația blocurilor / platforme
* Locația muniției / gloanțelor folosite de Chowder și de Gorgonzola
* Bara de viață a personajelor
* Inventarul personajelor

#region //Initializare

………………………………………………..

PictureBox[] bullet = new PictureBox[10000];

int bullet\_number = 0;

bool bullet\_active = false;

int direction = 1;

int[] bullet\_direction = new int[10000];

………………………………………………..

#endregion

1. *Subprogramul* „**void resolution(int x, int y)**” verifică rezoluția monitorului și verifică prin intermediul inițializării dacă această rezoluție diferă de rezoluția de bază, 1440 x 900.

…………………………………………..

while (contor != 1)

{

while (a != b) if (a > b) a -= b; else b -= a;

if (a > 1) { x /= a; y /= a; }

contor = a;

a = x; b = y;

}

…………………………………………..

1. *Subprogramul* „**void if\_resolution\_differs()**”, în cazul în care numărul de pixeli ai calculatorului, deci rezoluția, diferă de 1440 x 900, rezoluția se va schimba direct proporțional, astfel încât să se păstreze proporțiile și pe noul ecran.

…………………………………………………………

player.Height = y\_ax \* player.Height / y\_ax\_original;

player.Width = x\_ax \* player.Width / x\_ax\_original;

player.Top = y\_ax \* player.Top / y\_ax\_original;

player.Left = x\_ax \* player.Left / x\_ax\_original;

player.SizeMode = PictureBoxSizeMode.StretchImage;

ground = player.Top;

…………………………………………………………

1. *Subprogramul* „**void if\_resolution\_differs\_add\_player\_02()**”, la fel ca și programul precedent, modifică dimensiunile celui de-al 2-lea player în cazul adăugării sale.

……………………………….………………………….

player\_02.Height = y\_ax \* player\_02.Height / y\_ax\_original;

player\_02.Width = x\_ax \* player\_02.Width / x\_ax\_original;

player\_02.Top = y\_ax \* player\_02.Top / y\_ax\_original;

player\_02.Left = x\_ax \* player\_02.Left / x\_ax\_original;

player\_02.SizeMode = PictureBoxSizeMode.StretchImage;

ground\_02 = solid\_ground.Top - player\_02.Height;

……………………………….………………………….

1. *Subprogramul* „**private void Form1\_KeyDown(object sender, KeyEventArgs e)**” verifică dacă sunt apăsate butoanele necesare pentru controlarea personajului al cărui nume se regăsește și în titlul jocului, Chowder, și acționează în consecință.

…………………………………………………………....

if (e.KeyCode == Keys.D) // deplasare DREAPTA

{

right = true;

if (jump == false)

//player.Image = Image.FromFile("chowder\_right.png");

player.Image = Properties.Resources.chowder\_right;

direction = 1;

}

…………………………………………………………....

1. *Subprogramul* „**void add\_player\_02()**” adaugă cel de-al 2-lea personaj, Gorgonzola, pentru a fi controlat de un alt jucător și pentru a-l încurca pe Chowder.

……………………………………………………………….

player\_02 = new PictureBox();

player\_02.Height = 64;

player\_02.Width = 41;

player\_02.Top = 376;

player\_02.Left = 1300;

//player\_02.Image = Image.FromFile("gorgonzola\_left.png");

player\_02.Image = Properties.Resources.gorgonzola\_left;

Controls.Add(player\_02);

……………………………………………………………….

1. *Subprogramul* „**private void Form1\_KeyUp(object sender, KeyEventArgs e)**” verifică dacă butoanele „A” sau „D” pentru Chowder și „LEFT” sau „RIGHT” pentru Gorgonzola sunt apăsate, iar, dacă nu, deplasarea caracterului în stânga sau dreapta se va opri.

……………………………………………………………..

if (e.KeyCode == Keys.A)

{

left = false;

if (jump == false)

//player.Image = Image.FromFile("chowder\_small.png");

player.Image = Properties.Resources.Chowder\_small;

}

……………………………………………………………..

1. *Subprogramul* „**void right\_left\_move\_player\_02()**” este folosit la deplasarea lui Gorgonzola

……………………………………………………………

else if (left\_02 == true) //move left

{

player\_02.Left -= 5;

//player\_02.Image = Image.FromFile("gorgonzola\_left.png");

player\_02.Image = Properties.Resources.gorgonzola\_left;

}

……………………………………………………………

1. *Subprogramul* „**void jumping\_player\_02()**” verifică daca al 2-lea player, respecitv Gorgonzola, sare.

………………………………………

if (jump\_02 == true)

{

player\_02.Top -= Force\_02;

Force\_02 -= 1;

}

if (player\_02.Top >= ground\_02)

{

player\_02.Top = ground\_02;

jump\_02 = false;

}

else player\_02.Top += 5;

………………………………………

1. *Subprogramul* „**void collision\_01\_player\_02(PictureBox block)**” verifică dacă Gorgonzola se află pe o platformă iar, în caz pozitiv, nu cade de pe acel bloc.

……………………………………………………………………………………………….

if (player\_02.Left + player\_02.Width - 5 > block.Left) // blocked. bottom - limita

if (player\_02.Left + player\_02.Width + 5 < block.Left + block.Width + player\_02.Width)

if (player\_02.Top <= block.Bottom) // or k = 1;

if (player\_02.Top > block.Top)

{

k = 1; // or if (player.Top <= block.Bottom+10);

//player\_02.Image = Image.FromFile("chowder\_small.png");

jump\_02 = false;

Force\_02 = 0;

jump\_02 = true;

}

……………………………………………………………………………………………….

1. *Subprogramul* „**void collision\_02\_player\_02(PictureBox block)**” este, la fel ca și subprogramul precedent, menit să verifice limitele unei platforme.

………………………………………………………………………………………………..

if (player\_02.Left + player\_02.Width - 5 > block.Left) // blocked. top - limita

if (player\_02.Left + player\_02.Width + 5 < block.Left + block.Width + player\_02.Width)

if (player\_02.Top + player\_02.Height >= block.Top)

if (player\_02.Top < block.Top)

{

player\_02.Top = block.Top - player\_02.Height; // block.top = block.location.Y

jump\_02 = false;

// Force = 0;

}

………………………………………………………………………………………………..

1. *Subprogramul* „**void bullet\_moving\_player\_02()**” deplasează „bullet-ul” folosit și verifică dacă acesta lovește ceva.

………………………………………………………………………………………………

if (bullet\_direction\_02[i] == 1)

{

if (bullet\_02[i].Bounds.IntersectsWith(block.Bounds)) bullet\_02[i].Left = 3000;

else if (bullet\_02[i].Bounds.IntersectsWith(block\_02.Bounds)) bullet\_02[i].Left = 3000;

else if (bullet\_02[i].Bounds.IntersectsWith(block\_03.Bounds)) bullet\_02[i].Left = 3000;

else if (bullet\_02[i].Bounds.IntersectsWith(platform.Bounds)) bullet\_02[i].Left = 3000;

}

………………………………………………………………………………………………

1. *Subprogramul* „**void collect\_player\_02(PictureBox item, int count)**” verifică la al câtelea spațiu din inventar s-a ajuns iar fructul se va pune în primul spațiu gol al inventarului.

…………………………………………………………….………………………………….

else if (ticket\_inventory\_02 == 3)

{

if (count == 1) inventory\_03\_player\_02.Image = Properties.Resources.inventory\_apple;

else inventory\_03\_player\_02.Image = Properties.Resources.inventory\_banana;

if (Lifebar\_player\_02.Value + 20 < 100) Lifebar\_player\_02.Value += 20; //life increases

else Lifebar\_player\_02.Value = 100;

}

…………………………………………………………….………………………………….

1. *Subprogramul* „**void enemy\_collision\_player\_02(PictureBox enemy)**” verifică dacă personajele s-au intersectat cu Panini, iar în caz afirmativ, viața lor va scădea.

…………………………………………………….……………………………………..

if (player\_02.Bounds.IntersectsWith(enemy.Bounds))

{

panini\_text.Text = "Saruta-ma, Chowder !!!";

panini\_hittime = 1;

if (lovit > 100)

{

lovit = 0;

if (Lifebar\_player\_02.Value - 25 > 0)

{

Lifebar\_player\_02.Value -= 25;

life\_player\_02\_text.Text = Convert.ToString(Lifebar\_player\_02.Value + " %");

}

}

}

……………………………………………………….…………………………………..

1. *Subprogramul* „**void enemy\_02\_collision\_player\_02(PictureBox enemy)**” verifică dacă personajele nu s-au intersectat cu Shnitzel, iar în caz afirmativ, viața lor va scădea.

……………………………………………………………….

Lifebar\_player\_02.Value = 0;

life\_player\_02\_text.Text = Convert.ToString("0 %");

jump\_02 = true;

Force\_02 = G;

ground += 2000;

inchide = 490;

dieonce++;

if (dieonce == 1) MessageBox.Show("Gorgonzola a pierdut");

this.Close();

Cursor.Show();

menu main\_menu = new menu();

main\_menu.Show();

…………………………………………………………………

1. *Subprogramul* „**private void timer1\_Tick(object sender, EventArgs e)**” verifică de mai multe ori în fiecare secundă dacă evenimentele menționate se petrec, acționând în cauză.

………………………………

collision\_01(block);

collision\_01(block\_02);

collision\_01(block\_03);

jumping();

collect(food\_1, 1); // 1 - mar ; 2 - banana

collect(food\_2, 2);

collect(food\_3, 1);

collect(food\_4, 2);

………………………………

1. *Subprogramul* „**void right\_left\_move()**” verifică dacă Chowder se deplasează în dreapta sau în stânga

……………………………………………………………..

else if (left == true) //move left

{

player.Left -= 5;

if (jump == false)

player.Image = Properties.Resources.chowder\_left;

//player.Image = Image.FromFile("chowder\_left.png");

}

……………………………………………………………..

1. *Subprogramul* „**void bullet\_moving()**” ajută la deplasarea „bullet-ului” într-o direcție și verifică dacă lovește ceva.

……………………………………………………………………………………....

if (bullet\_direction[i] == 1)

{

if (bullet[i].Bounds.IntersectsWith(block.Bounds)) bullet[i].Left = 3000;

else if (bullet[i].Bounds.IntersectsWith(block\_02.Bounds)) bullet[i].Left = 3000;

else if (bullet[i].Bounds.IntersectsWith(block\_03.Bounds)) bullet[i].Left = 3000;

else if (bullet[i].Bounds.IntersectsWith(platform.Bounds)) bullet[i].Left = 3000; }

……………………………………………………………………………………….

1. *Subprogramul* „**void jumping()**”, în momentul apăsării tastei „space”, realizează saltul.

……………………………………………………….

if (player.Top >= ground)

{

player.Top = ground;

if (jump == true)

//player.Image = Image.FromFile("chowder\_small.png");

player.Image = Properties.Resources.Chowder\_small;

jump = false;

}

……………………………………………………….

1. *Subprogramul* „**void collision\_01(PictureBox block)**” verifică dacă player 1, adică Chowder, se lovește de platforme.

…………………………………………………………………………………..

if (player.Left + player.Width - 5 > block.Left) // blocked. bottom - limita

if (player.Left + player.Width + 5 < block.Left + block.Width + player.Width)

if (player.Top <= block.Bottom) // or k = 1;

if (player.Top > block.Top)

{

k = 1; // or if (player.Top <= block.Bottom+10);

//player.Image = Image.FromFile("chowder\_small.png");

player.Image = Properties.Resources.Chowder\_small;

jump = false;

Force = 0;

jump = true;

}

…………………………………………………………………………………..

1. *Subprogramul* „**void collision\_02(PictureBox block)**” face același lucru ca și programul precedent, însă, spre deosebire de el, verifică limita față de partea de jos a blocului.

………………………………………………………………………….

if (jump == true) player.Image = Properties.Resources.Chowder\_small;

player.Top = block.Top- player.Height; // block.top = block.location.Y

jump = false;

// Force = 0;

…………………………………………………………………………..

1. *Subprogramul* „**void collect(PictureBox item, int count)**” colectează fructele și le așează în inventar.

……………………………………………………….

if (player.Bounds.IntersectsWith(item.Bounds))

if (Convert.ToString(item.Tag) == Convert.ToString(0))

ticket\_inventory++;

…………………………………………………………….

1. *Subprogramul* „**void enemy\_move(PictureBox enemy, int lft, int rgt)**” verifică dacă primul player, Chowder, este atins de Panini, caz în care un procentaj din viața sa va fi luat.

……………………………………………………………………..

if (enemy.Left <= lft) miscare = 3;

if (enemy.Right >= rgt) miscare = -3;

enemy.Left += miscare;

panini\_text.Left = enemy.Left;

if (miscare == 3) enemy.Image = Properties.Resources.panini\_right;

if (miscare == -3) enemy.Image = Properties.Resources.panini\_left;

……………………………………………………………………..

1. *Subprogramul* „**void enemy\_move\_02(PictureBox enemy, int lft, int rgt)**” verifică dacă primul player, Chowder, este atins de Shnitzel, caz în care un procentaj din viața sa va fi luat.

…………………………………………………………..

if (lovit > 100)

{

lovit = 0;

if (Lifebar.Value - 25 > 0)

{

Lifebar.Value -= 25;

life\_text.Text = Convert.ToString(Lifebar.Value + " %");

}

}

…………………………………………………………..

1. *Subprogramul* „**void MungDaal\_finish()**” verifică dacă Chowder are toate ingredientele pentru a-i permite sa treacă nivelul.

………………………………………….

if (mung\_finish == true) tick++;

if (tick == 100)

{

timer1.Stop();

MessageBox.Show("Chowder a castigat");

Cursor.Show();

this.Close();

menu main\_menu = new menu();

main\_menu.Show();

} // dupa 3 secunde se inchide

………………………………………….

1. *Subprogramul* „**void platform\_move(PictureBox floating, int up, int down)**” deplasează platforma în sus și în jos, sau, în funcție de nivel, în stânga și în dreapta.

……………………………………….......

if (floating.Top <= up) miscare\_03 = 3;

if (floating.Top >= down) miscare\_03 = -3;

floating.Top += miscare\_03;

collision\_01(platform);

collision\_02(platform);

……………………………………….......

1. *Subprogramul* „**void limits(PictureBox person)**” bordează jocul pentru a nu putea fi parcurs în exteriorul marginii ecranului.

………………………………………………………………………….

if (person.Right > limit\_right) person.Left = limit\_right - person.Width;

if (person.Left < 0) person.Left = 0;

………………………………………………………………………….

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* Meniul principal al jocului conține și el, la rândul lui, secvențe de cod pentru a putea face trecerea de la interfața respectivă la nivelul propriu-zis. Algoritmii utilizați sunt:
* Determinarea C.M.M.D.C. / în funcție de pixeli
* Schimbarea rezoluției în funcție de rezoluția monitorului
* Adăugarea sunetului pentru fiecare nivel în parte
* Închiderea meniului principal
* Caseta care apare înainte ca jocul să fie lansat în mod „fulscreen” pe ecran, cu rol informativ, conține o secvență scurtă de cod cu proprietatea de închidere a aplicației în urma apăsării butonului „Cancel” sau de intrare în meniu prin intermediul butonului „Continue”. De asemenea, este fereastra în care se afișează și LOGO-ul meu, realizat în Photoshop.
* În urma ieșirii din joc, pe ecran va apărea o imagine care va fi înghețată timp de 3 secunde, fără posibilitatea de a trece peste, după care se va închide, astfel jocul fiind oprit complet, incluzând toate procesele acestuia.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# 4. CONCLUZII

 Pe tot parcursul realizării acestui proiect, am acumulat un număr mare de informații noi și folositoare pentru o viitoare carieră de programator. Încet, dar sigur, cunoștințele noi sunt clădite pe fundația informațiilor din trecut, astfel, pot spune că acest joc simplist, chiar primitiv din punctul unora de vedere, m-a ajutat într-o mare masură. În acest fel m-am obisnuit cu programarea de tip „object-oriented”, orientată pe obiecte, dar și cu o interfață nouă de programare, astfel usurându-mi viitoarele necesități de adaptare în ceea ce privește limbajul de programare și interfața folosită. Sunt de părere că toate lucrurile învățate până în clasa a XII-a referitoare la Visual C# vor servi drept suport pentru studiul individual ce va urma, indubitabil, sa vină.

~ Îmbunătățiri ~

Aplicația constituie un instrument de divertisment, plasând jucătorul în mijlocul acțiunii. Asemenea oricărui alt joc prezent pe piață, sunt o gramadă de lucruri care pot fi adăugate ulterior pentru o mai bună experiență personală:

* Putem realiza o interfață grafică mai „populată”, având un număr mai mare de elemente prezente în „background”.
* Putem adăuga elemente noi precum: armuri, muniție mai puternică, bombe sau alte obiecte, desigur, cu scopul de a ajuta jucătorul.
* Putem adăuga mai multe nivele, viariabile, diferite, mai mari și mai dificile.
* Putem introduce dificultăți noi, astfel oferim posibilitatea jucătorului de a-și alege singur greutatea de trecere a nivelului.
* Putem adăuga un număr mai mare de caractere ce pot fi controlate, astfel se adaugă opțiunea de „multiplayer” iar jocul devine mai invens.
* Putem adăuga inamici noi cu abilități speciale.
* Putem adăuga inamici având aceeași abilitate ca și caracterele controlate de jucător, și anume abilitatea de a trage cu muniție.
* Putem introduce opțiunea de „leveling up”, însemnând deblocarea nivelelor succesive în urma completării nivelelor anterioare.
* Putem introduce o hartă care se derulează pe măsură ce o parcurgem, astfel putem să ne dezvoltăm pe orizontală.
* Putem să punem un „best time” pentru fiecare mapă, astfel forțându-te să îți dobori recordul de fiecare dată.
* De asemenea, putem introduce o platformă online, în acest fel oferindu-se posibilitatea de „online multiplayer”.

# 5. BIBLIOGRAFIE

* *Aplicații Windows în Visual C# 2008 Express Edition*

**Editura POLIROM**

Ana Întuneric / Cristina Sichim / Daniela Tarasă

* *INFORMATICĂ pentru LICEU și BACALAUREAT*

**Colecția Donaris - Info**

Pavel Florin Moraru

* *INFORMATICĂ C++ manual pentru clasa a XI –a*

**Editura GIMNASIUM**

Sandra Junea / Adriana Simulescu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~