CREAZIONE DI VIDEOGIOCHI CON C# E WINDOWS FORM

Autori: Franco Sicca e chatGPT

NOTA: Alcuni riferimenti nelle spiegazioni potrebbero avere nomi in italiano o in inglese mentre nel codice i nomi potrebbero essere invertiti: quindi bisogna leggere criticamente la spiegazione (come è sempre buona norma fare:-)

[**ARKANOID 3**](#_50dohgf5n8k)

[STRUTTURA DEL GIOCO 3](#_oa813ibwchqv)

[📂 Struttura del Codice 3](#_659eb7ot0xcg)

[1. Form principale (ArkanoidForm) 3](#_j6pa8633hubg)

[2. Timer di gioco 3](#_277y5g6mleus)

[3. Disegno grafico (Paint) 4](#_c8kfmzju8kkd)

[4. Controlli da tastiera 4](#_h7r2nnpfabmz)

[CODICE COMMENTATO 4](#_cdk76xjgpjwl)

[✅ Program.cs 4](#_a031wgkshul)

[✅ GameForm.cs 4](#_jxgrf7l2ynny)

[PILLOLE DI TEORIA 7](#_mqitfqecedym)

[Cos'è l'evento Paint? 7](#_nrvm9y6eee4o)

[🧠 Cosa fa this.Paint += Disegna;? 8](#_yqw3kvvthduu)

[🧪 Esempio pratico 8](#_a6nkujeo99p8)

[🎯 Perché è importante? 8](#_1vtsgxl2695k)

[**SNAKE 9**](#_3z3xj0e0169o)

[STRUTTURA DEL GIOCO 9](#_5i84sfiwu50a)

[🧠 1. Logica Principale del Gioco 9](#_5vqgo4ike7fb)

[Oggetti fondamentali: 9](#_ewco2e3oev7l)

[🔁 2. Ciclo di gioco (Timer.Tick) 9](#_poc9v38e92wv)

[CODICE 10](#_804b8hh16bc8)

[PILLOLE DI TEORIA 13](#_1r074eyn9sdg)

[Aumentare la velocità progressivamente 13](#_40q88jfw53kr)

[Parte Grafica – Paint e Invalidate() 13](#_h2e3tideq58n)

[this.Paint += Disegna; 13](#_u8d1ao5icyxz)

[Invalidate() 14](#_usdfxm428iew)

[Uso delle Immagini (PictureBox o Graphics.DrawImage) 14](#_qzigxoidzkg0)

[Opzione 1 – Graphics.DrawImage 14](#_wcgfy5ucmrbg)

[Opzione 2 – PictureBox 15](#_6e9rm768y0th)

[Esempio di ciclo grafico con immagini 15](#_rmhvzl3m5zsh)

[🎯 In sintesi 15](#_1k5vpg26k86b)

[**DINO CHROME 17**](#_lqbpv3wo89rs)

[STRUTTURA DEL GIOCO 17](#_w3k8nzv45u2n)

[🎮 Come si gioca: 17](#_be6rl2o80lew)

[CODICE 17](#_6z4ub4oetow8)

[🔹 1. Program.cs (entry point standard) 17](#_3ez2ujtbpi42)

[🔹 2. DinoForm.cs (logica e grafica del gioco) 18](#_iikp5ovht70g)

[PILLOLE DI TEORIA 21](#_f2xmgqq2gj6t)

[🎮 Controlli: Su e Giù 21](#_23o4uzn8ecew)

[🎯 1. Freccia SU (Salto) 21](#_bs4qgely7tww)

[🔧 Implementazione: 21](#_8ycfbsyc8n78)

[🧠 Cosa succede a ogni Tick del Timer: 22](#_pp9p983z16a5)

[🎯 2. Freccia GIÙ (Abbassarsi) 22](#_18b4qnrown1v)

[🔧 Implementazione: 22](#_g5wlntfoodk3)

[✍️ Riepilogo del Ciclo di Gioco 23](#_1pejoblzmk5d)

[🎨 Parte grafica 23](#_t7er0yva5tr2)

[🎨 Personalizzazione: 24](#_ppuxvs3b3az)

[Per usare immagini: 24](#_uumy2re6p0co)

[📂 Struttura del Codice 24](#_kscxj2hzqwf4)

[1. Form principale (DinoForm) 24](#_wehnfzo2afv2)

[2. Timer di gioco 25](#_fogud57hc7a4)

[3. Disegno grafico (Paint) 25](#_j4pe5v8epbfh)

[4. Controlli da tastiera 25](#_dxrj2cd2h979)

[📌 Punti Chiave 25](#_d73r7bgaiktc)

[**CODICE DEI TRE GIOCHI COMMENTATO PER EVIDENZIRE LA PROGRAMMAZIONE AD OGGETTI 26**](#_1rgxvl8hmrr5)

[ARKANOID 26](#_e04o4vp0hj39)

[SNAKE 31](#_6od6m1eco4wo)

[DINO CHROME 33](#_euvxe9ai4gu4)

# ARKANOID

Ecco il **codice completo del gioco in C# stile Arkanoid**, con paddle, palla, rimbalzi e mattoni. È un progetto base Windows Forms per la programmazione di videogiochi in C#.

## STRUTTURA DEL GIOCO

Gioco di Arkanoid utilizza una classe Graphics per disegnare l'ambiente grafico del gioco e utilizza unaclasse Timer per aggiornare la grafica ogni intervallo di tempo specificato

### **📂 Struttura del Codice**

#### **1. Form principale (ArkanoidForm)**

* Inizia impostando dimensioni e attivando DoubleBuffered = true per evitare sfarfallii grafici.
* Contiene i principali elementi del gioco:  
  + Rectangle paddle: la barra del giocatore.
  + Rectangle ball: la pallina.
  + Point ballVelocity: velocità della pallina.
  + Rectangle[] bricks: array di mattoni.

#### **2. Timer di gioco**

* Imposta un intervallo di 20ms.
* Ad ogni Tick cioè ad ogni intervallo di tempo impostato, viene richiamato il metodo Aggiorna()
* Nel metodo Aggiorna() :  
  + Rileva collisioni con pareti, barra e mattoni.
  + Ferma il gioco se la pallina cade in basso.
  + Aggiorna posizione della pallina.
    - // Richiede il ridisegno
    - Invalidate();
  + Il metodo Invalidate() richiama l'evento Paint ed quindi il metodo associato Disegna()

#### **3. Disegno grafico (Paint)**

* Usa Graphics.FillRectangle e Graphics.FillEllipse per disegnare:  
  + Barra, pallina e mattoni.

#### **4. Controlli da tastiera**

* Frecce **Sinistra** e **Destra** per spostare la barra.

## CODICE COMMENTATO

### **✅ Program.cs**

using System;

using System.Windows.Forms;

namespace ArkanoidBase

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new GameForm());

}

}

}

### **✅ GameForm.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace ArkanoidBase

{

public class GameForm : Form

{

private Timer timer; // Timer per il ciclo di gioco

private Rectangle paddle; // Racchetta del giocatore

private Rectangle ball; // Palla del gioco

private List<Rectangle> bricks; // Lista dei mattoni

private int ballDX = 4; // Direzione orizzontale della palla

private int ballDY = -4; // Direzione verticale della palla

private const int paddleSpeed = 10; // Velocità della racchetta

private bool sinistra, destra; // Tasti premuti per movimento

public GameForm()

{

this.DoubleBuffered = true;

this.Width = 800;

this.Height = 600;

this.Text = "Arkanoid Base";

this.KeyPreview = true;

// Collega gli eventi

this.Paint += Disegna;

this.KeyDown += TastoPremuto;

this.KeyUp += TastoRilasciato;

// Imposta e avvia il timer

timer = new Timer();

timer.Interval = 20;

timer.Tick += Aggiorna;

timer.Start();

// Inizializza la racchetta e la palla

paddle = new Rectangle(350, 550, 100, 15);

ball = new Rectangle(390, 300, 20, 20);

InizializzaMattoni();

}

private void InizializzaMattoni()

{

bricks = new List<Rectangle>();

// Crea 5 righe e 10 colonne di mattoni

for (int r = 0; r < 5; r++)

{

for (int c = 0; c < 10; c++)

{

bricks.Add(new Rectangle(60 + c \* 65, 50 + r \* 25, 60, 20));

}

}

}

private void Disegna(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

// Disegna la racchetta

g.FillRectangle(Brushes.Blue, paddle);

// Disegna la palla

g.FillEllipse(Brushes.Red, ball);

// Disegna i mattoni

foreach (var brick in bricks)

{

g.FillRectangle(Brushes.Green, brick);

g.DrawRectangle(Pens.Black, brick);

}

}

private void Aggiorna(object sender, EventArgs e)

{

// Movimento della racchetta

if (sinistra && paddle.X > 0) paddle.X -= paddleSpeed;

if (destra && paddle.X + paddle.Width < ClientSize.Width) paddle.X += paddleSpeed;

// Movimento della palla

ball.X += ballDX;

ball.Y += ballDY;

// Rimbalzi ai bordi

if (ball.X < 0 || ball.X + ball.Width > ClientSize.Width) ballDX = -ballDX;

if (ball.Y < 0) ballDY = -ballDY;

// Rimbalzo sulla racchetta

if (ball.IntersectsWith(paddle)) ballDY = -Math.Abs(ballDY);

// Rimbalzo sui mattoni

for (int i = 0; i < bricks.Count; i++)

{

if (ball.IntersectsWith(bricks[i]))

{

bricks.RemoveAt(i);

ballDY = -ballDY;

break;

}

}

// Game over se la palla cade sotto

if (ball.Y > ClientSize.Height)

{

timer.Stop();

MessageBox.Show("Game Over!");

Application.Exit();

}

// Richiede il ridisegno

Invalidate();

}

private void TastoPremuto(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Left) sinistra = true;

if (e.KeyCode == Keys.Right) destra = true;

}

private void TastoRilasciato(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Left) sinistra = false;

if (e.KeyCode == Keys.Right) destra = false;

}

}

}

## PILLOLE DI TEORIA

### **Cos'è l'evento Paint?**

L'evento Paint viene **generato ogni volta che il form o un controllo deve essere ridisegnato**:

* all'avvio dell'applicazione,
* quando viene ridimensionato,
* quando è coperto e poi scoperto,
* o quando viene chiamato manualmente Invalidate().

### **🧠 Cosa fa this.Paint += Disegna;?**

* this.Paint è l'evento del form.
* += Disegna significa: "aggiungi il metodo Disegna come gestore di questo evento".

### **🧪 Esempio pratico**

Supponiamo che tu abbia questo metodo:

private void Disegna(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

g.FillRectangle(Brushes.Blue, new Rectangle(10, 10, 100, 50));

}

Con this.Paint += Disegna; stai dicendo al form:

"Ogni volta che devi ridisegnarti, esegui anche il metodo Disegna".

Il metodo riceve:

* sender → il form stesso,
* PaintEventArgs e → contiene e.Graphics, l’oggetto per disegnare sullo schermo.

### **🎯 Perché è importante?**

Ti permette di **disegnare dinamicamente** elementi come:

* paddle, pallina e mattoni in un gioco,
* grafici personalizzati,
* interfacce custom.

# SNAKE

Vediamo la **struttura del gioco Snake** in C# con **Windows Forms**, introducendo anche:

* 🧱 Come è organizzato il gioco
* 🖼️ Il disegno grafico con Invalidate()
* 🎨 Come sostituire elementi disegnati con immagini

## STRUTTURA DEL GIOCO

### **🧠 1. Logica Principale del Gioco**

#### **Oggetti fondamentali:**

* **Snake**: una lista di rettangoli (List<Rectangle> o List<Point>) che rappresentano le singole “celle” del corpo.
* **Cibo**: un rettangolo o punto generato casualmente nella griglia.
* **Direzione**: un enum o string per memorizzare il movimento ("up", "down", "left", "right").
* **Timer**: un oggetto System.Windows.Forms.Timer che scandisce l’avanzamento del gioco.

### **🔁 2. Ciclo di gioco (Timer.Tick)**

Ad ogni tick del timer:

1. **Sposta il serpente** (aggiunge un nuovo elemento nella direzione attuale e rimuove l'ultimo).
2. **Verifica collisioni**:  
   * Con se stesso
   * Con i bordi
   * Con il cibo
3. **Aggiorna lo stato**:  
   * Se ha mangiato il cibo, **non rimuove** l'ultimo elemento → il serpente cresce.
   * Altrimenti, mantiene la lunghezza costante.

## CODICE

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace SnakeGame

{

public partial class SnakeForm : Form

{

// Attributi principali (campi) del gioco

private List<Point> snake = new List<Point>(); // La lista che rappresenta il corpo del serpente

private Point food; // La posizione del cibo

private string direction = "right"; // Direzione iniziale del movimento

private Timer gameTimer; // Timer per il movimento automatico

private int gridSize = 20; // Dimensione delle celle della griglia

private Random rand = new Random(); // Generatore di numeri casuali

public SnakeForm()

{

InitializeComponent();

this.Width = 400;

this.Height = 400;

// Imposta gli eventi del Form

this.Paint += SnakeForm\_Paint; // Evento Paint → chiamato ogni volta che va ridisegnato il form

this.KeyDown += SnakeForm\_KeyDown; // Evento KeyDown → per gestire la tastiera

StartGame(); // Avvia il gioco

}

private void StartGame()

{

snake.Clear();

snake.Add(new Point(5, 5)); // Crea il punto iniziale del serpente

direction = "right";

SpawnFood();

// Inizializza e configura il timer

gameTimer = new Timer();

gameTimer.Interval = 200; // Intervallo di aggiornamento (più basso = più veloce)

gameTimer.Tick += GameTick; // Evento Tick → gestisce il movimento del serpente

gameTimer.Start();

}

private void SpawnFood()

{

// Posiziona il cibo in un punto casuale

food = new Point(rand.Next(0, this.ClientSize.Width / gridSize),

rand.Next(0, this.ClientSize.Height / gridSize));

}

// Evento Tick del Timer → aggiornamento posizione serpente

private void GameTick(object sender, EventArgs e)

{

Point head = snake[0]; // La testa del serpente

Point newHead = head;

// Calcola la nuova posizione della testa in base alla direzione

switch (direction)

{

case "up": newHead.Y -= 1; break;

case "down": newHead.Y += 1; break;

case "left": newHead.X -= 1; break;

case "right": newHead.X += 1; break;

}

// Controllo collisione con bordi

if (newHead.X < 0 || newHead.Y < 0 ||

newHead.X >= this.ClientSize.Width / gridSize ||

newHead.Y >= this.ClientSize.Height / gridSize ||

snake.Contains(newHead)) // collisione con se stesso

{

gameTimer.Stop(); // Ferma il gioco

MessageBox.Show("Game Over");

StartGame(); // Riavvia il gioco

return;

}

snake.Insert(0, newHead); // Aggiungi la nuova testa

// Se il serpente mangia il cibo

if (newHead == food)

{

SpawnFood(); // Genera nuovo cibo

}

else

{

snake.RemoveAt(snake.Count - 1); // Altrimenti rimuove la coda per mantenere la lunghezza

}

this.Invalidate(); // Richiede il ridisegno → genera l’evento Paint

}

// Evento Paint → disegna serpente e cibo

private void SnakeForm\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

// Disegna il cibo

g.FillEllipse(Brushes.Red, food.X \* gridSize, food.Y \* gridSize, gridSize, gridSize);

// Disegna il corpo del serpente

for (int i = 0; i < snake.Count; i++)

{

Brush b = (i == 0) ? Brushes.Green : Brushes.LightGreen; // testa e corpo

g.FillRectangle(b, snake[i].X \* gridSize, snake[i].Y \* gridSize, gridSize, gridSize);

}

}

// Evento KeyDown → cambia direzione

private void SnakeForm\_KeyDown(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Up:

if (direction != "down") direction = "up";

break;

case Keys.Down:

if (direction != "up") direction = "down";

break;

case Keys.Left:

if (direction != "right") direction = "left";

break;

case Keys.Right:

if (direction != "left") direction = "right";

break;

}

}

}

}

## PILLOLE DI TEORIA

## **Aumentare la velocità progressivamente**

Come diminuire la velocità del gioco **aumentando l’intervallo del timer**? Nel codice, trovi questa riga:

gameTimer.Interval = 150;

Questo significa che ogni 150 millisecondi si aggiorna la posizione del serpente. Per rallentare il gioco, **aumenta il numero**. Ad esempio:

gameTimer.Interval = 250; // più lento

Oppure per renderlo molto lento all’inizio:

gameTimer.Interval = 400;

📌 Suggerimento: se vuoi una **difficoltà progressiva**, puoi diminuire gradualmente il valore di gameTimer.Interval man mano che il punteggio aumenta.

## **Parte Grafica – Paint e Invalidate()**

### **this.Paint += Disegna;**

* In Windows Forms, il disegno avviene tramite il metodo Paint, che riceve un oggetto Graphics.

Dentro a Disegna(object sender, PaintEventArgs e) usi:  
  
 csharp  
CopiaModifica  
e.Graphics.FillRectangle(Brushes.Green, rettangolo);

### **Invalidate()**

* Ogni volta che vuoi **ridisegnare** la finestra, chiami Invalidate(), che forza una nuova esecuzione di Paint.
* Viene normalmente chiamato nel metodo Tick del timer.

## **Uso delle Immagini (PictureBox o Graphics.DrawImage)**

Puoi **sostituire gli elementi disegnati (pallini, rettangoli)** con immagini:

### **Opzione 1 – Graphics.DrawImage**

Nel metodo Paint, invece di:

csharp

CopiaModifica

e.Graphics.FillRectangle(Brushes.Green, segment);

puoi usare:

csharp

CopiaModifica

Image snakeHead = Image.FromFile("snakehead.png");

e.Graphics.DrawImage(snakeHead, headRect);

Puoi anche mantenere riferimenti in memoria per evitare di ricaricare le immagini ad ogni tick.

### **Opzione 2 – PictureBox**

Meno flessibile per un gioco in tempo reale, ma più facile da usare:

* Crea dinamicamente dei PictureBox con .Image = ... e .Location = ... per ogni segmento del serpente.
* Aggiorna le posizioni ad ogni tick.

⚠️ **Sconsigliato per serpenti lunghi o gioco veloce**, perché meno performante rispetto a Graphics.

## **Esempio di ciclo grafico con immagini**

csharp

CopiaModifica

Image headImage = Image.FromFile("head.png");

Image bodyImage = Image.FromFile("body.png");

private void Disegna(object sender, PaintEventArgs e)

{

for (int i = 0; i < snake.Count; i++)

{

Rectangle r = new Rectangle(snake[i].X \* cellSize, snake[i].Y \* cellSize, cellSize, cellSize);

if (i == 0)

e.Graphics.DrawImage(headImage, r);

else

e.Graphics.DrawImage(bodyImage, r);

}

Rectangle foodRect = new Rectangle(food.X \* cellSize, food.Y \* cellSize, cellSize, cellSize);

e.Graphics.DrawImage(Image.FromFile("apple.png"), foodRect);

}

## **🎯 In sintesi**

|  |  |
| --- | --- |
| **Componente** | **Ruolo** |
| Timer | Gestisce il tempo e gli aggiornamenti |
| Invalidate() | Richiede il ridisegno della finestra |
| Paint | Ridisegna serpente e cibo |
| List<Point> | Rappresenta il corpo del serpente |
| DrawImage() | Inserisce immagini al posto dei rettangoli |

# DINO CHROME

Controllare un **dinosauro** che salta per evitare **ostacoli** in arrivo.

## STRUTTURA DEL GIOCO

Il clone di Dino Chrome è un **gioco endless runner 2D**:

* Il giocatore controlla un dinosauro che corre da solo.
* Lo scopo è **saltare gli ostacoli** (es. cactus) o **abbassarsi** per evitare altri ostacoli (es. uccelli, se previsti).

Il gioco è basato su:

* Un **timer** che aggiorna continuamente lo stato del gioco.
* Il metodo Paint che disegna tutto.
* La gestione della **fisica del salto**.
* La gestione dei **tasti premuti**.

### 🎮 Come si gioca:

* **FRECCIA SU** → il dinosauro salta.
* Evita i rettangoli marroni (ostacoli).
* Il gioco termina al primo impatto.

## CODICE

### **🔹 1. Program.cs (entry point standard)**

csharp

CopiaModifica

using System;

using System.Windows.Forms;

namespace DinoChrome

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new DinoForm());

}

}

}

### **🔹 2. DinoForm.cs (logica e grafica del gioco)**

csharp

CopiaModifica

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace DinoChrome

{

public class DinoForm : Form

{

private Timer gameTimer = new Timer();

private Rectangle dino;

private List<Rectangle> obstacles = new List<Rectangle>();

private int gravity = 2;

private int velocity = 0;

private int jumpStrength = -20;

private bool jumping = false;

private int groundY = 150;

private Random rand = new Random();

private int score = 0;

private Font font = new Font("Consolas", 12);

private int obstacleSpeed = 5;

public DinoForm()

{

this.Width = 600;

this.Height = 250;

this.DoubleBuffered = true;

this.Text = "Dino Chrome in C#";

this.BackColor = Color.White;

dino = new Rectangle(50, groundY, 40, 50);

gameTimer.Interval = 20;

gameTimer.Tick += GameTick;

gameTimer.Start();

this.Paint += Draw;

this.KeyDown += OnKeyDown;

this.KeyUp += OnKeyUp;

}

private void GameTick(object sender, EventArgs e)

{

if (jumping)

{

dino.Y += velocity;

velocity += gravity;

if (dino.Y >= groundY)

{

dino.Y = groundY;

jumping = false;

velocity = 0;

}

}

// Muove ostacoli

for (int i = 0; i < obstacles.Count; i++)

{

obstacles[i] = new Rectangle(obstacles[i].X - obstacleSpeed, obstacles[i].Y, obstacles[i].Width, obstacles[i].Height);

}

// Rimuove ostacoli usciti dallo schermo

obstacles.RemoveAll(o => o.X + o.Width < 0);

// Crea nuovi ostacoli

if (rand.Next(0, 100) < 3)

{

obstacles.Add(new Rectangle(this.Width, groundY + 20, 20, 40));

}

// Collisione

foreach (Rectangle obs in obstacles)

{

if (dino.IntersectsWith(obs))

{

gameTimer.Stop();

MessageBox.Show($"Hai perso! Punteggio: {score}");

Application.Exit();

}

}

score++;

Invalidate();

}

private void Draw(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

g.FillRectangle(Brushes.Black, dino);

foreach (Rectangle obs in obstacles)

{

g.FillRectangle(Brushes.Brown, obs);

}

g.DrawString("Punteggio: " + score, font, Brushes.DarkGreen, 10, 10);

}

private void OnKeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Up && !jumping)

{

jumping = true;

velocity = jumpStrength;

}

}

private void OnKeyUp(object sender, KeyEventArgs e)

{

// Espandibile: ad esempio abbassarsi col tasto giù

}

}

}

## PILLOLE DI TEORIA

### 🎮 Controlli: Su e Giù

#### 🎯 1. Freccia SU (Salto)

Quando il giocatore preme la freccia su (Keys.Up), il dinosauro deve **saltare**.

### **🔧 Implementazione:**

csharp

CopiaModifica

private bool jumping = false;

private int velocity = 0;

private int gravity = 2;

private int jumpStrength = -20;

private int groundY = 150; // Y di base

private void Form\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Up && !jumping)

{

jumping = true;

velocity = jumpStrength; // Parte il salto verso l’alto

}

}

##### 🧠 Cosa succede a ogni Tick del Timer:

csharp

CopiaModifica

private void GameTimer\_Tick(object sender, EventArgs e)

{

if (jumping)

{

dino.Y += velocity; // Muove il dinosauro in verticale

velocity += gravity; // Simula accelerazione verso il basso (gravità)

// Se raggiunge il terreno, ferma il salto

if (dino.Y >= groundY)

{

dino.Y = groundY;

jumping = false;

velocity = 0;

}

}

Invalidate(); // Ridisegna la scena

}

#### 🎯 2. Freccia GIÙ (Abbassarsi)

Nel gioco originale di Chrome, il dinosauro si **abbassa** con la freccia giù per evitare ostacoli volanti (se presenti).  
 Nel clone puoi simularlo **riducendo l’altezza** del dinosauro temporaneamente.

### **🔧 Implementazione:**

csharp

CopiaModifica

private bool ducking = false;

private void Form\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Down && !jumping)

{

ducking = true;

dino.Height = 30; // si abbassa

dino.Y = groundY + 20;

}

}

private void Form\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Down)

{

ducking = false;

dino.Height = 50; // torna normale

dino.Y = groundY;

}

}

### ✍️ Riepilogo del Ciclo di Gioco

|  |  |  |
| --- | --- | --- |
| **Azione** | **Codice** | **Effetto** |
| **Freccia SU** | velocity = jumpStrength | Avvia il salto |
| **Durante il salto** | Y += velocity, velocity += gravity | Sale e poi scende come un vero salto |
| **Frequenza** | Regolata dal Timer.Interval | Velocità di aggiornamento |
| **Freccia GIÙ** | Height -=, Y += | Il dinosauro si abbassa |

### 🎨 Parte grafica

Il dinosauro viene disegnato in Paint:

csharp

CopiaModifica

private void Form\_Paint(object sender, PaintEventArgs e)

{

e.Graphics.FillRectangle(Brushes.Black, dino);

}

Può essere sostituito con un'immagine:

csharp

CopiaModifica

Image dinoImg = Image.FromFile("dino.png");

e.Graphics.DrawImage(dinoImg, dino);

### 🎨 Personalizzazione:

#### Per usare immagini:

Sostituisci nel metodo Draw:

csharp

CopiaModifica

g.FillRectangle(Brushes.Black, dino);

con:

csharp

CopiaModifica

Image dinoImg = Image.FromFile("dino.png");

g.DrawImage(dinoImg, dino);

E fai lo stesso per gli ostacoli.

#### 📂 Struttura del Codice

#### **1. Form principale (DinoForm)**

* Finestra 600x200, DoubleBuffered attivo.
* Elementi principali:  
  + Rectangle dino: il personaggio.
  + List<Rectangle> obstacles: lista di ostacoli generati.
  + Variabili gravity, velocity, jumping, score.

#### **2. Timer di gioco**

* Ogni 20ms aggiorna il gioco:  
  + Se jumping == true, modifica dino.Y simulando un salto.
  + Sposta gli ostacoli a sinistra.
  + Genera nuovi ostacoli casuali.
  + Controlla collisioni tra dinosauro e ostacoli.
  + Aumenta il punteggio.

#### **3. Disegno grafico (Paint)**

* Disegna il dinosauro (nero) e gli ostacoli (marroni).
* Mostra il punteggio in alto a sinistra.

#### **4. Controlli da tastiera**

* Barra spaziatrice (Space) per saltare, solo se non è già in aria.

#### 📌 Punti Chiave

* La simulazione del salto è realistica: salita con velocity, discesa con gravity.
* Gli ostacoli si muovono e vengono eliminati quando escono dallo schermo.
* Il gioco termina al primo impatto con un ostacolo.

# CODICE DEI TRE GIOCHI COMMENTATO PER EVIDENZIRE LA PROGRAMMAZIONE AD OGGETTI

## ARKANOID

// -----------------------------------------------------------------------------

// ArkanoidForm.cs (Commentato)

// Un semplice clone di Arkanoid che dimostra concetti di POO in C#.

//

// ▶ Ereditarietà: ArkanoidForm DERIVA da System.Windows.Forms.Form

// ▶ Attributi (campi): paddle, ball, ballDirection, bricks, score, timer

// ▶ Metodi: costruttore, MovePaddle (handler evento KeyDown), GameTick (handler

// evento Timer.Tick), Draw (handler evento Paint)

// ▶ Eventi: KeyDown, Paint, Tick

// -----------------------------------------------------------------------------

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Arkanoid

{

// ArkanoidForm eredita da Form ⇒ raccoglie tutte le funzionalità di una finestra

public class ArkanoidForm : Form

{

// -------------------- Attributi --------------------

private Timer timer = new Timer(); // Evento Tick per il game‑loop

private Rectangle paddle; // Barra controllata dal giocatore

private Rectangle ball; // Pallina

private Point ballDirection = new Point(4, -4); // Velocità (dx,dy) della pallina

private Rectangle[] bricks; // Array di mattoni

private int score = 0; // Punteggio

// ---------------- Costruttore ---------------------

public ArkanoidForm()

{

// Proprietà ereditate da Form

this.DoubleBuffered = true; // Evita flickering

this.Width = 600;

this.Height = 400;

this.Text = "Arkanoid – OOP Demo";

// Inizializza attributi

paddle = new Rectangle(250, 350, 100, 10);

ball = new Rectangle(290, 330, 20, 20);

// Crea 3 righe × 10 colonne di mattoni

bricks = new Rectangle[30];

int idx = 0;

for (int r = 0; r < 3; r++)

for (int c = 0; c < 10; c++)

bricks[idx++] = new Rectangle(10 + c \* 58, 30 + r \* 25, 55, 20);

// ----- Eventi -----

timer.Interval = 20;

timer.Tick += GameTick; // Evento Tick → metodo GameTick

timer.Start();

this.Paint += Draw; // Evento Paint → metodo Draw

this.KeyDown += MovePaddle; // Evento KeyDown → metodo MovePaddle

}

// --------------- Metodi / Event Handlers ----------------

// Sposta la barra in risposta all'evento KeyDown

private void MovePaddle(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Left && paddle.Left > 0)

paddle.X -= 20;

else if (e.KeyCode == Keys.Right && paddle.Right < ClientSize.Width)

paddle.X += 20;

}

// Game‑loop: si attiva ad ogni Tick del Timer

private void GameTick(object sender, EventArgs e)

{

// Aggiorna posizione pallina

ball.X += ballDirection.X;

ball.Y += ballDirection.Y;

// Collisione con pareti

if (ball.Left <= 0 || ball.Right >= ClientSize.Width)

ballDirection.X = -ballDirection.X;

if (ball.Top <= 0)

ballDirection.Y = -ballDirection.Y;

// Collisione con barra

if (ball.IntersectsWith(paddle))

ballDirection.Y = -ballDirection.Y;

// Collisione con mattoni

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

{

if (bricks[i] != Rectangle.Empty && ball.IntersectsWith(bricks[i]))

{

bricks[i] = Rectangle.Empty; // “Distrugge” il mattone

ballDirection.Y = -ballDirection.Y;

score++;

break;

}

}

// Game over: la pallina esce in basso

if (ball.Bottom >= ClientSize.Height)

{

timer.Stop();

MessageBox.Show("Game Over!");

Application.Exit();

}

Invalidate(); // Richiede ridisegno (evento Paint)

}

// Disegna tutti gli elementi di gioco

private void Draw(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

g.FillRectangle(Brushes.Blue, paddle);

g.FillEllipse(Brushes.Red, ball);

foreach (var brick in bricks)

if (brick != Rectangle.Empty)

g.FillRectangle(Brushes.Green, brick);

g.DrawString($"Score: {score}", new Font("Consolas", 12), Brushes.Black, 10, 10);

}

}

}

## SNAKE

// -----------------------------------------------------------------------------

// SnakeForm.cs (Commentato)

// Implementazione di Snake con struttura OOP.

//

// ▶ Attributi: timer, snake (List<Point>), direction, food, cellSize, rand

// ▶ Metodi: costruttore, GameTick, Draw, OnKeyDown, GenerateFood

// ▶ Eventi: Timer.Tick, Paint, KeyDown

// -----------------------------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace SnakeGame

{

public class SnakeForm : Form

{

// ---------------- Attributi ----------------

private Timer timer = new Timer(); // Game‑loop

private List<Point> snake = new List<Point>(); // Corpo del serpente

private Point direction = new Point(1, 0); // Direzione iniziale (destra)

private Point food; // Cibo

private int cellSize = 20; // Dimensione griglia

private Random rand = new Random();

private bool gameOver = false;

private Font font = new Font("Consolas", 12);

// --------------- Costruttore ---------------

public SnakeForm()

{

Width = 400;

Height = 400;

DoubleBuffered = true;

Text = "Snake – OOP Demo";

// Inizia con un segmento

snake.Add(new Point(5, 5));

GenerateFood();

timer.Interval = 150;

timer.Tick += GameTick;

timer.Start();

Paint += Draw;

KeyDown += OnKeyDown;

}

// ------------ Game‑loop -----------------

private void GameTick(object sender, EventArgs e)

{

if (gameOver) return;

// Crea nuova testa in base alla direzione

Point newHead = new Point(snake[0].X + direction.X, snake[0].Y + direction.Y);

// Controllo collisioni con bordi o se stesso

if (newHead.X < 0 || newHead.Y < 0 ||

newHead.X >= ClientSize.Width / cellSize ||

newHead.Y >= ClientSize.Height / cellSize ||

snake.Contains(newHead))

{

gameOver = true;

timer.Stop();

MessageBox.Show("Game Over!");

Application.Exit();

}

snake.Insert(0, newHead); // Aggiunge la nuova testa

if (newHead == food) // Ha mangiato?

GenerateFood(); // Non rimuove la coda → cresce

else

snake.RemoveAt(snake.Count - 1); // Mantiene lunghezza

Invalidate(); // Ridisegna

}

// --------------- Disegno ----------------

private void Draw(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

// Corpo del serpente

foreach (Point p in snake)

g.FillRectangle(Brushes.Green, p.X \* cellSize, p.Y \* cellSize, cellSize, cellSize);

// Cibo

g.FillEllipse(Brushes.Red, food.X \* cellSize, food.Y \* cellSize, cellSize, cellSize);

g.DrawString($"Length: {snake.Count}", font, Brushes.Black, 5, 5);

}

// --------------- Input -----------------

private void OnKeyDown(object sender, KeyEventArgs e)

{

// Evita movimento inverso immediato

if (e.KeyCode == Keys.Up && direction.Y != 1) direction = new Point(0, -1);

else if (e.KeyCode == Keys.Down && direction.Y != -1) direction = new Point(0, 1);

else if (e.KeyCode == Keys.Left && direction.X != 1) direction = new Point(-1, 0);

else if (e.KeyCode == Keys.Right && direction.X != -1) direction = new Point(1, 0);

}

// --------------- Cibo ------------------

private void GenerateFood()

{

int maxX = ClientSize.Width / cellSize;

int maxY = ClientSize.Height / cellSize;

do

{

food = new Point(rand.Next(maxX), rand.Next(maxY));

} while (snake.Contains(food));

}

}

}

## DINO CHROME

// -----------------------------------------------------------------------------

// DinoForm.cs (Commentato)

// Endless‑runner stile Dino Chrome.

// Mostra salti, gravità e gestione eventi tastiera.

//

// ▶ Ereditarietà: DinoForm : Form

// ▶ Attributi: timer, dino, obstacles, gravity, etc.

// ▶ Metodi: costruttore, GameTick, Draw, OnKeyDown

// ▶ Eventi: Timer.Tick, Paint, KeyDown

// -----------------------------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace DinoChrome

{

public class DinoForm : Form

{

// ---------------- Attributi ----------------

private Timer gameTimer = new Timer(); // Game‑loop

private Rectangle dino; // Player

private List<Rectangle> obstacles = new List<Rectangle>(); // Ostacoli

private int gravity = 2; // Accelerazione verso il basso

private int velocity = 0; // Velocità verticale corrente

private int jumpStrength = -20; // Impulso di salto

private bool jumping = false; // Stato salto

private int groundY = 150; // Y terreno

private Random rand = new Random();

private int score = 0;

private Font font = new Font("Consolas", 12);

private int obstacleSpeed = 5;

// --------------- Costruttore ----------------

public DinoForm()

{

Width = 600;

Height = 250;

DoubleBuffered = true;

Text = "Dino Chrome – OOP Demo";

BackColor = Color.White;

dino = new Rectangle(50, groundY, 40, 50);

// Wiring eventi

gameTimer.Interval = 20;

gameTimer.Tick += GameTick;

gameTimer.Start();

Paint += Draw;

KeyDown += OnKeyDown;

}

// ----------- Game‑loop -----------------

private void GameTick(object sender, EventArgs e)

{

// Gestione salto con “fisica” semplificata

if (jumping)

{

dino.Y += velocity;

velocity += gravity;

if (dino.Y >= groundY)

{

dino.Y = groundY;

jumping = false;

velocity = 0;

}

}

// Muove gli ostacoli

for (int i = 0; i < obstacles.Count; i++)

obstacles[i] = new Rectangle(obstacles[i].X - obstacleSpeed, obstacles[i].Y, obstacles[i].Width, obstacles[i].Height);

obstacles.RemoveAll(o => o.Right < 0);

// Genera nuovi ostacoli

if (rand.Next(0, 100) < 3)

obstacles.Add(new Rectangle(ClientSize.Width, groundY + 20, 20, 40));

// Collisione

foreach (var obs in obstacles)

if (dino.IntersectsWith(obs))

{

gameTimer.Stop();

MessageBox.Show($"Game Over – Score: {score}");

Application.Exit();

}

score++;

Invalidate(); // Chiede ridisegno

}

// ------------- Disegno ----------------

private void Draw(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

g.FillRectangle(Brushes.Black, dino);

foreach (Rectangle obs in obstacles)

g.FillRectangle(Brushes.Brown, obs);

g.DrawString($"Score: {score}", font, Brushes.DarkGreen, 10, 10);

}

// ------------- Input ------------------

private void OnKeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Up && !jumping)

{

jumping = true;

velocity = jumpStrength; // Impulso iniziale verso l'alto

}

}

}

}