

download

ATOMIC - LINGUAGGIO DI PROGRAMMAZIONE DIDATTICO

Francesco Cortesi
Founder of BerGAME



BerGAME

 **SORINT** lab

DISCLAIMER

Atomic è un progetto nuovo.

Non essendoci ancora studi e dati su larga scala, la maggior parte delle informazioni **non tecniche** riportate nella presentazione sono **considerazioni personali** frutto della mia esperienza personale come sviluppatore e formatore e del confronto con docenti della scuola primaria.

[download](#)



COME È NATO IL PROGETTO

BerGAME sviluppa Serious Game: videogiochi educativi.



BerGAME: Fiera per l'infanzia **Lilliput** (2015)

[download](#)



SORINT lab

COME È NATO IL PROGETTO

Atomic risponde a questi problemi degli insegnanti:

- È completamente in Italiano
- È un linguaggio testuale
- È graficamente semplice, minimale, non dispersivo
- Non richiede accesso a internet

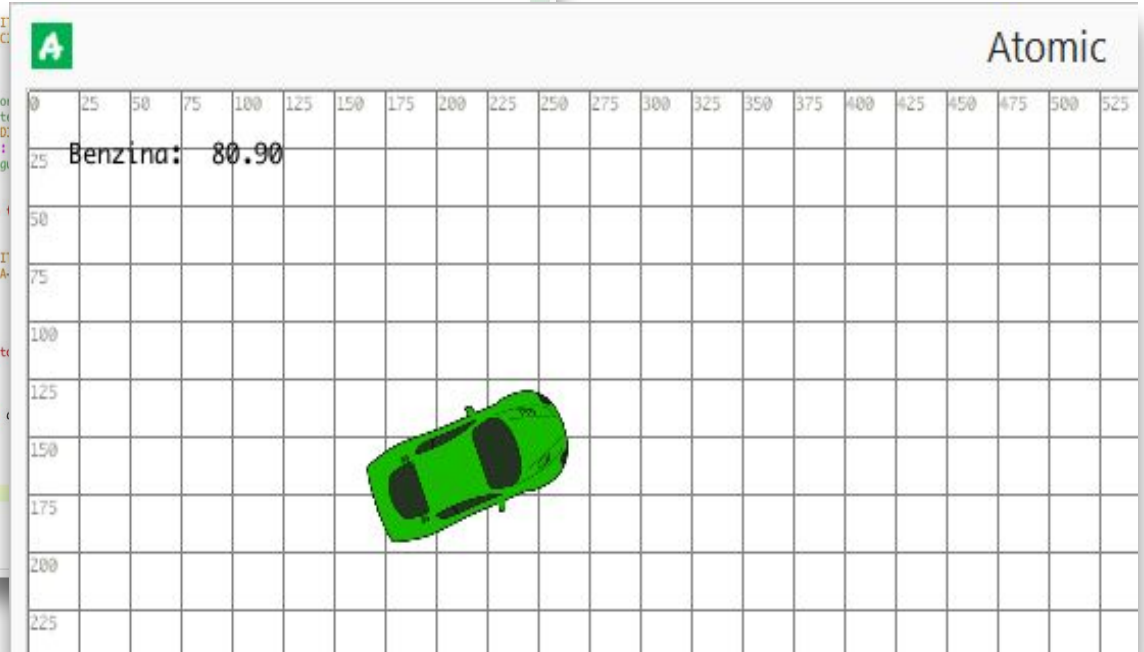
[download](#)




```

Atomic
1 INIZIA
2 foto = ottieni immagine => (NOME:"internet/images.vectorhq.com/images/previews/be6/green-racing-car-top-view-104981.png") (ORIGINE X: 246.5) (ORIGINE Y: 122.5)
3 crea un esemplare => (NOME: auto) (IMMAGINE: foto) (SCALA ASSE X: 0.2) (SCALA ASSE Y: 0.2) (benzina: 100) (accelerazione: 0.5) (velocità_massima: 20)
4
5 CICLO CONTINUO
6 //Accelerazione, freno e retromarcia
7 se benzina del auto è maggiore di 0
8 {
9 se tasto freccia su è premuto = vero { modifica un elemento => (NOME: auto) (VELOCITA: VELOCITA + 0.5)
10 se tasto freccia giù è premuto = vero { modifica un elemento => (NOME: auto) (VELOCITA: VELOCITA - 0.5)
11 }
12
13 //Sterzo
14 velocità_assoluta = ottieni il valore assoluto di => (VALORE: VELOCITA del auto) //Questo valore è sempre positivo
15 //La direzione è proporzionale alla velocità: più l'auto è veloce, più l'auto sterza velocemente
16 se tasto freccia destra è premuto = vero { modifica un elemento => (NOME: auto) (DIREZIONE: DIREZIONE + 1)
17 se tasto freccia sinistra è premuto = vero { modifica un elemento => (NOME: auto) (DIREZIONE: DIREZIONE - 1)
18 modifica un elemento => (NOME: auto) (ROTAZIONE: DIREZIONE) //La rotazione dell'immagine è uguale alla direzione
19
20 //Disegna la benzina attuale
21 testo = ottieni testo combinato => (TESTO 1: "Benzina: ") (TESTO 2: benzina del auto) disegna il testo
22
23 //Decelerazione per attrito
24 se VELOCITA del auto è maggiore di 0 { modifica un elemento => (NOME: auto) (VELOCITA: VELOCITA * 0.9)
25 se VELOCITA del auto è minore di 0 { modifica un elemento => (NOME: auto) (VELOCITA: VELOCITA * 0.9)
26
27 //semplificazione fisica per fermare l'auto
28 se tasto freccia su è premuto = falso e tasto freccia giù è premuto = falso
29 {
30 se VELOCITA del auto è minore di 1 e VELOCITA del auto è maggiore di 1*-1 { modifica un elemento => (NOME: auto) (VELOCITA: 0)
31 }
32
33 //Limiti di velocità
34 se VELOCITA del auto è maggiore di velocità_massima del auto { modifica un elemento => (NOME: auto) (VELOCITA: velocità_massima del auto)
35 se VELOCITA del auto è minore di -6 { modifica un elemento => (NOME: auto) (VELOCITA: -6)
36
37 //Teletrasporto
38 trasporta elemento al lato opposto quando esce dalla finestra => (ELEMENTO: auto)
39

```



download



PERCHÉ USARE UN LINGUAGGIO TESTUALE?



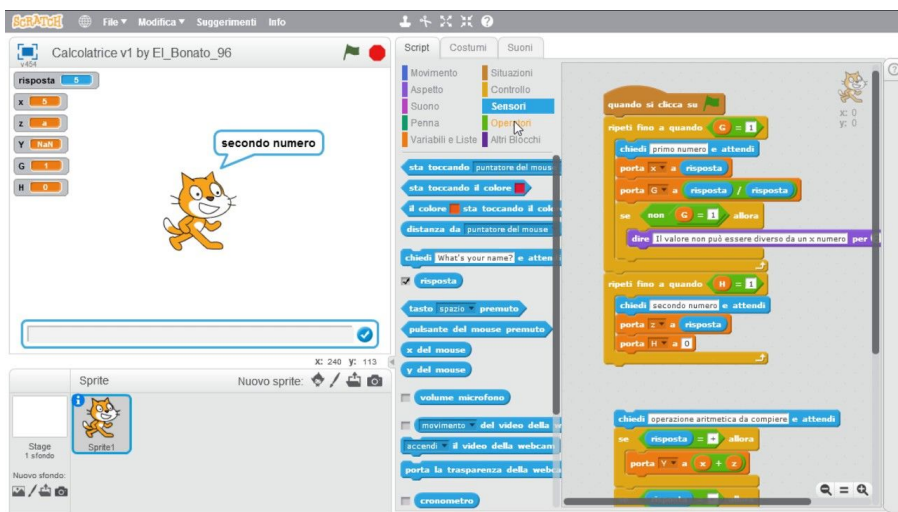
Scratch e code.org utilizzano la programmazione visuale.

I linguaggi visuali non permettono di commettere errori sintattici.

[download](#)



PERCHÉ USARE UN LINGUAGGIO TESTUALE?



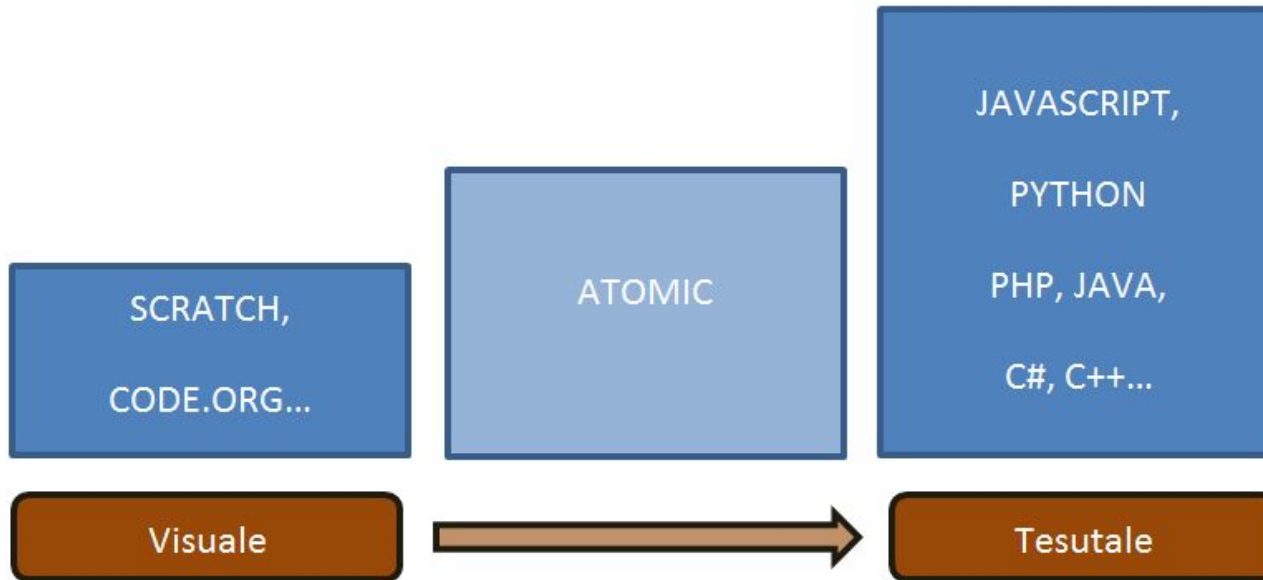
La programmazione testuale permette, invece, di apprendere la sintassi di un linguaggio.

Non basta più incastrare in modo intuitivo elementi pre-esistenti, bisogna confrontarsi con la paura del **foglio bianco** (letteralmente!).

[download](#)



PERCHÉ USARE ATOMIC?



È il “**Gradino mancante**” nella scala dell’apprendimento della programmazione.

PERCHÉ IN ITALIANO?



- La propria lingua madre è la migliore per imparare **concetti nuovi**.
- Non è scontato avere una buona conoscenza dell'**inglese** in **età evolutiva**.
- Anche code.org e Scratch hanno una versione in italiano.

[download](#)





ILLUSTRAZIONE SINTETICA DELL'AMBIENTE DI SVILUPPO

[download](#)



MANIFESTO

Atomic è un linguaggio di programmazione a **scopo didattico**.

- Realmente **facile da imparare**, soprattutto per chi non ha mai programmato
- **Esplicito e intuitivo**, facilmente leggibile e **prossimo all'italiano parlato**
- **Flessibile, tollerante** ma preciso

- Non è fatto per creare progetti a lungo termine: è solo un linguaggio che permette di “partire a razzo” con la programmazione

[download](#)



IN PRATICA

Script didattici brevi (massimo 30-40 righe) ma **funzionanti e concludenti**.



```
1 INIZIA
2 //inserisci qui il codice da eseguire una sola volta all'inizio
3 ingrandisci = vero
4 raggio = 50
5 massimo = 55
6 minimo = 45
7
8 CICLO CONTINUO
9 //inserisci qui il codice da eseguire in continuazione
10 disegna cerchio → (RAGGIO: raggio) (COLORE: rosso) (X: larghezza finestra/2) (Y: altezza finestra/2)
11
12 se ingrandisci = vero allora aumenta raggio di 1 .
13 se ingrandisci = falso allora diminuisci raggio di 1 .
14 se raggio > massimo allora ingrandisci = falso .
15 se raggio < minimo allora ingrandisci = vero .
16
```

[download](#)

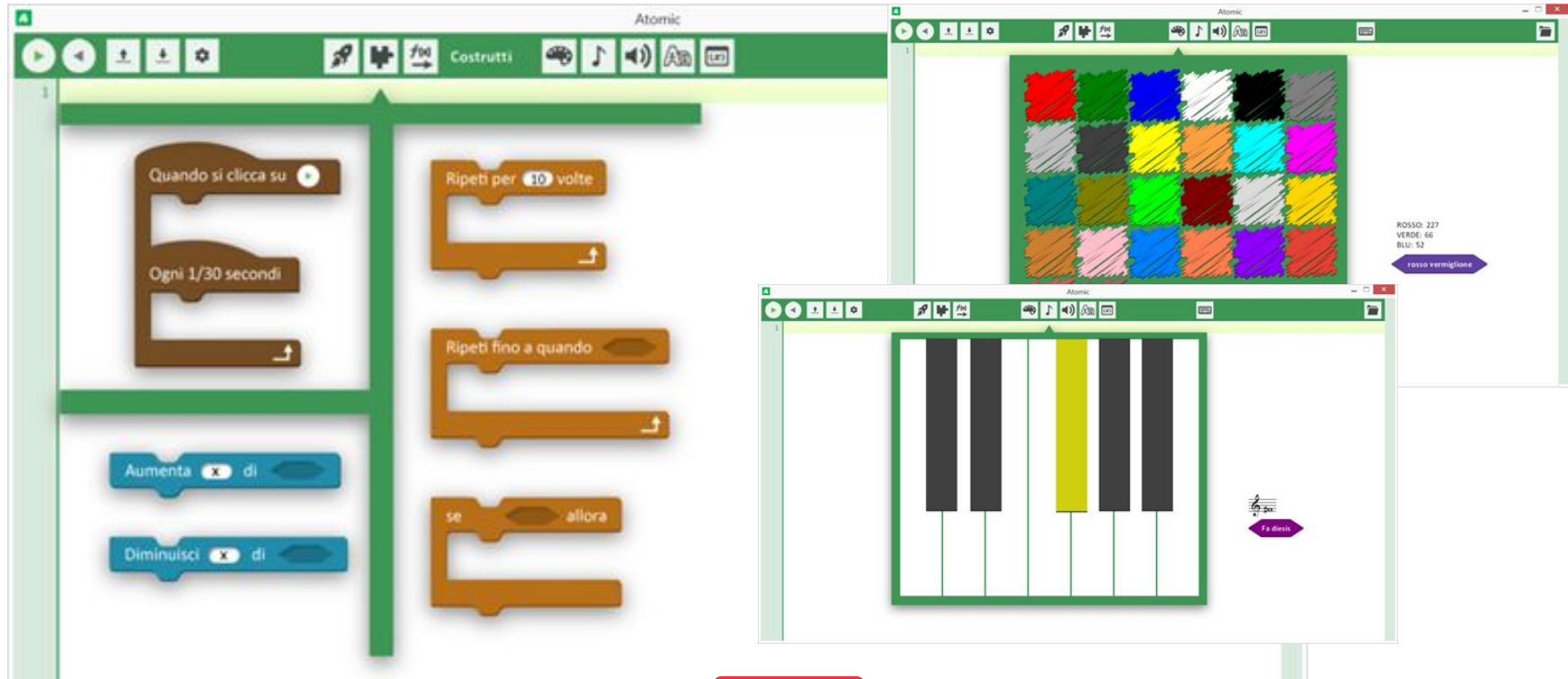


MINI EDITOR

- Le interfacce permettono di inserire **frammenti di codice** pronti all'uso
- Le interfacce si rifanno alla **programmazione a blocchi** (simili a quelle utilizzate in Scratch e Code.org)
- Il blocco selezionato viene **convertito in testo** e inserito nell'editor.



MINI EDITOR



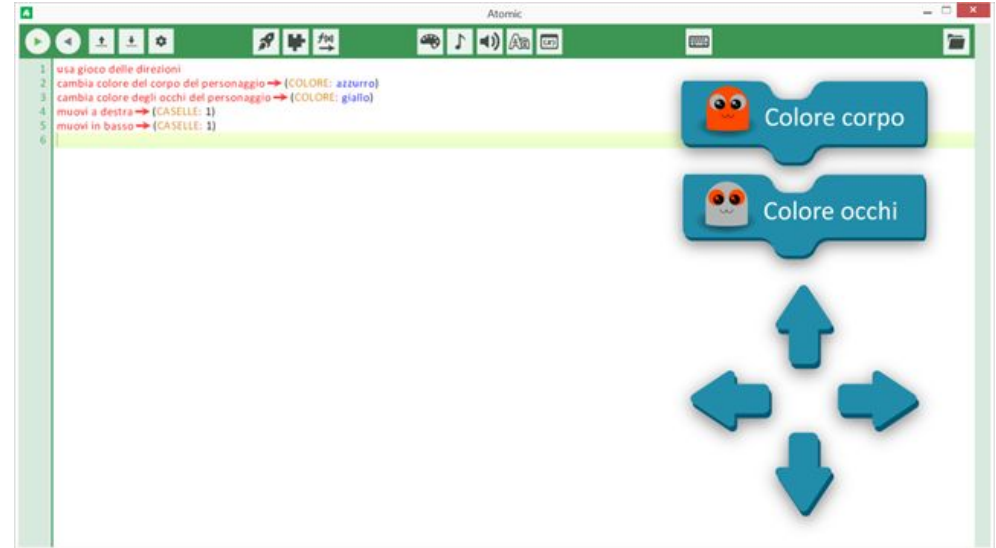
[download](#)



KIT PRONTI

L'icona **Kit pronti** permette di inserire pezzi di codice pronti per essere eseguiti e modificati.

Alcuni kit comprendono **funzioni speciali** e possono far comparire delle **interfacce aggiuntive**.



[download](#)



ILLUSTRAZIONE SINTETICA DEL LINGUAGGIO ATOMIC



[download](#)



TIPI DI DATI

In Atomic esistono solo due tipi di dato: **numero** e **testo**.



```
1 numero = 10
2 testo = "Ciao mondo!"
```

EVENTI

Esistono solo due eventi in Atomic: **INIZIA**(Start/Setup) e **CICLO CONTINUO**(Loop).

```
1  INIZIA
2  //inserisci qui il codice da eseguire una sola volta all'inizio
3
4  CICLO CONTINUO
5  //inserisci qui il codice da eseguire in continuazione
6
```

[download](#)



VARIABILI

nome = valore

- Nel corso della sua esistenza una variabile può cambiare il tipo di dato che contiene (tipizzazione dinamica).
- Esistono delle **variabili integrate** facili da usare (Es: **x del mouse**, **tasto invio è stato premuto**, **timer 1**, **colore sfondo...**)

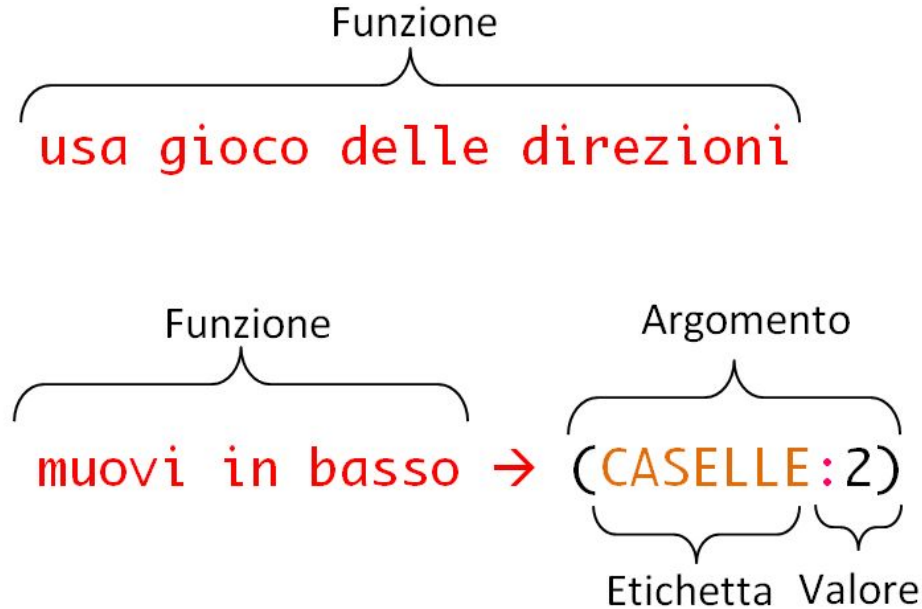
AUMENTA E DIMINUISCI

Per aumentare/diminuire in modo intuitivo il valore di una variabile:
aumenta e diminuisci.

```
1 | aumenta orsetto_lavatore di 1
```

```
1 | diminuisci orsetto_lavatore di 2
```

FUNZIONI



Gli **argomenti** (parametri) delle funzioni hanno un'**etichetta** e possono essere scritti in un **ordine casuale**.

FUNZIONI

Esempio:

```
1 | disegna cerchio → (X: 100) (Y: 300) (RAGGIO: 200)
```

che può anche essere scritta senza problemi in questo modo:

```
1 | disegna cerchio → (Y: 300) (RAGGIO: 200) (X: 100)
```

Inoltre è possibile aggiungere in modo chiaro altri argomenti supportati dalla funzione:

```
1 | disegna cerchio → (Y: 300) (RAGGIO: 200) (X: 100) (COLORE: blu)
```

[download](#)



FUNZIONI

è anche possibile non specificare degli argomenti (anche se fondamentali), ad esempio:

1 | **disegna cerchio** → (COLORE: blu)

disegnerà un cerchio blu di una dimensione imprecisata in un punto imprecisato.

1 | **disegna cerchio**

disegnerà un cerchio nero (colore di base) di una dimensione imprecisata in un punto imprecisato.

[download](#)



FUNZIONI “OTTIENI”

Tutte le funzioni che iniziano con “**ottieni**”

restituiscono un risultato che può essere memorizzato in una variabile:

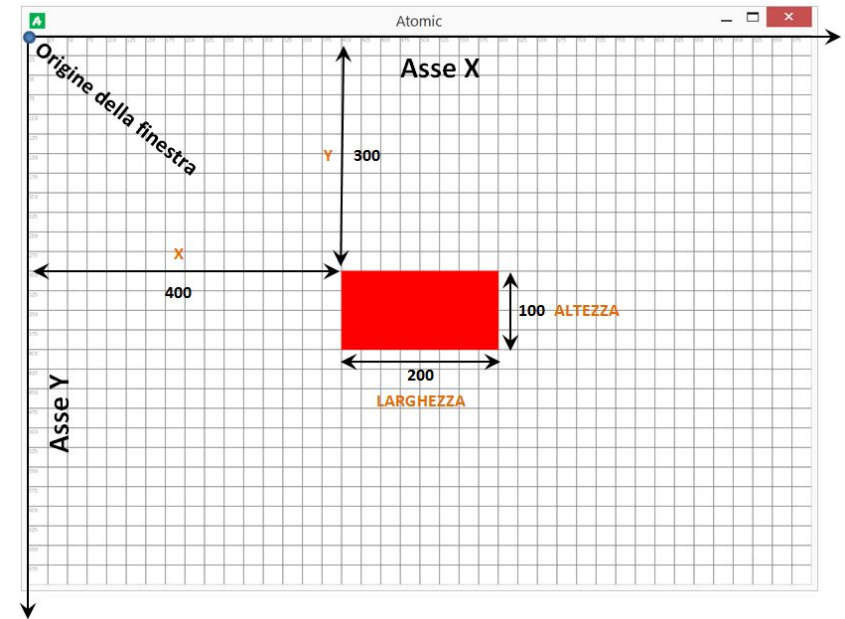
```
1 gatto = ottieni uno a caso di questi valori → (VALORE 1 : 50 ) (VALORE 2 : 100 ) (VALORE 3 : 70 )
2 gatto = ottieni il logaritmo naturale di → (VALORE : 324 )
3 mio_colore = ottieni uno a caso di questi valori → (VALORE 1 : verde ) (VALORE 2 : verde oliva )
```

[download](#)



CHE FUNZIONI SONO PRESENTI IN ATOMIC?

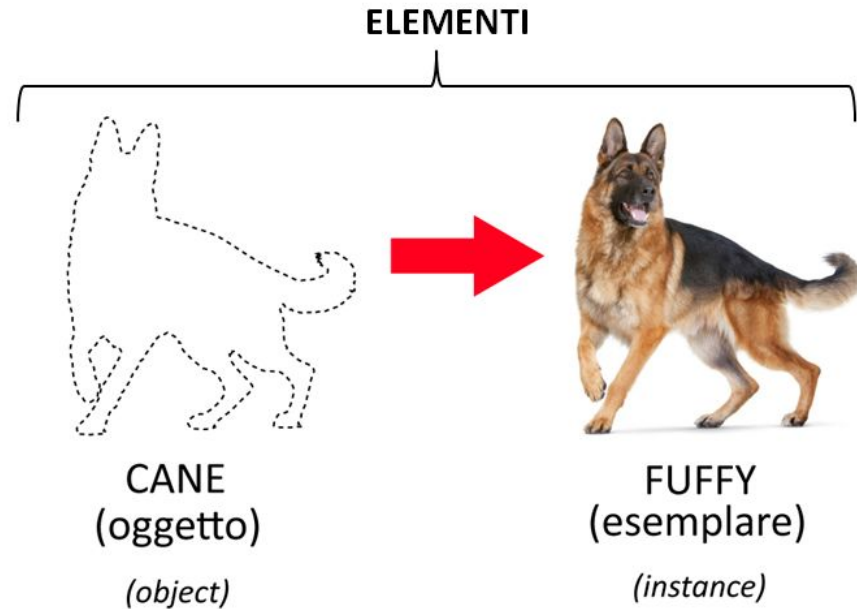
- Funzioni di **disegno** (forme geometriche, testo, immagini, colori)
- Funzioni di **casualità e matematiche**
- Funzioni per la manipolazione del **testo**
- Funzioni sull'**audio e musicali**
- Funzioni sulle **interfacce utente**
- Funzioni sugli **oggetti**
- e molte altre...



ELEMENTI: OGGETTI ED ESEMPLARI

Oggetto: *cane* Esempio: *Fuffy*

- L'oggetto *cane* è la descrizione generica di un cane.
- L'esemplare *Fuffy* è un esemplare di *cane*, unico, identificabile e concreto
- *Fuffy* e *cane* sono elementi.



CARATTERISTICHE DI ESEMPLARI E OGGETTI

- Gli oggetti/esemplari hanno delle **caratteristiche** integrate (principalmente grafiche e “fisiche”).
- Coincidono con le loro **variabili locali integrate** e con gli **argomenti delle funzioni** sugli oggetti.

```
1 immagine_freccia=ottieni immagine → (IMMAGINE: "immagini/freccia.png")
2
3 crea un oggetto → (NOME: freccia) // Nome dello oggetto
4 (X: 20) (Y: 20) // Coordinate in cui verranno creati i suoi esemplari
5 (DIREZIONE: -45) (ROTAZIONE: -45) // Direzione di movimento e rotazione dell'immagine
6 (VELOCITA: 3) // Velocità di movimento in pixel per step
7 (IMMAGINE: immagine_freccia) // L'immagine che rappresenta l'oggetto
```

VARIABILI LOCALI

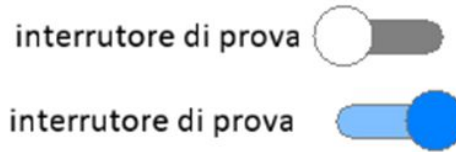
Ogni oggetto/esemplare oltre alle variabili locali integrate può contenere delle **variabili locali personalizzate**.

```
1 | crea un oggetto → (NOME: automobile) (benzina: 100)
```

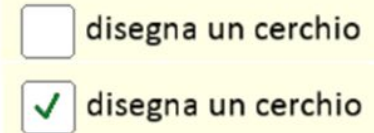
INTERFACCE PER CREARE APPLICAZIONI



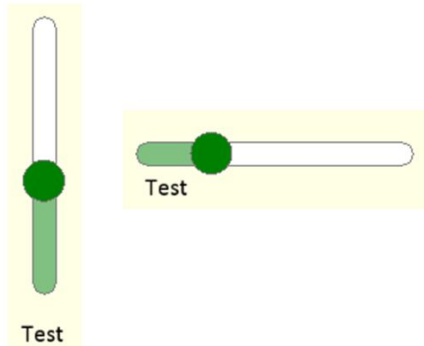
Tasti virtuali



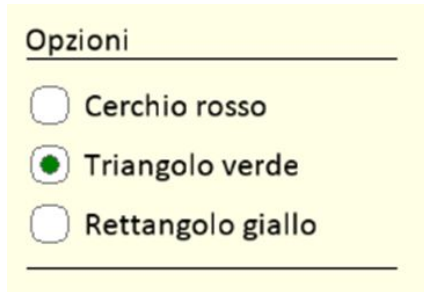
Interruttori



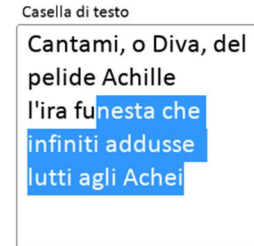
Caselle di spunta



Barre di controllo



Gruppi di opzioni



Caselle di testo

COSTRUTTI CONDIZIONALI E ITERATIVI

Costrutto **se** per istruzioni condizionali.

```
1 se gatto = 1 allora disegna cerchio → (RAGGIO: 200) (COLORE: blu).
```

Costrutti **finche** e **ripeti** per iterazioni.

```
1 INIZIA
2 cerchi=0
3
4 CICLO CONTINUO
5 cerchi=1
6 ripeti per 20 volte
7 {
8 disegna cerchio → (RAGGIO: 5*cerchi) (COLORE: rosso) (X: 40*cerchi)
9 aumenta cerchi di 1
10 }
```

TABELLE (ARRAYS)

- Monodimensionali o bidimensionali
- Utilizzabili tramite **funzioni specifiche**
- Per riferirsi alle **celle** di una **tabella** si usano **RIGHE** e **COLONNE**

	COLONNA 1	COLONNA 2	COLONNA 3
RIGA 1	1	4324	523
RIGA 2	32	21	65
RIGA 3	2	76	25
RIGA 4	432432	6347	6
RIGA 5	32.1	654	3



DEFINIRE NUOVE FUNZIONI

È possibile definire nuove funzioni usando il costrutto **definisci funzione**.

```
1  definisci funzione "ottieni area rettangolo"  
2  {  
3  area = <"BASE">*<"ALTEZZA">  
4  }  
5  con questa funzione ottieni area  
6  }  
7  
8  
9  area_rettangolo_a = ottieni area rettangolo → (BASE: 100) (ALTEZZA: 75)  
10 area_rettangolo_b = ottieni area rettangolo → (BASE: 20) (ALTEZZA: 35)  
11 area_rettangolo_c = ottieni area rettangolo → (BASE: 60) (ALTEZZA: 15)  
12 area_rettangolo_d = ottieni area rettangolo → (BASE: 700) (ALTEZZA: 25)  
13
```

INCLUDERE CODICE ESTERNO

È possibile includere codice esterno.

```
1 | includi "codici/esempi_facili/esempio.txt"
```

All'interno del codice esterno è possibile definire funzioni che potranno poi essere usate nel codice invocante.

IMPORTARE FUNZIONI ESTERNE TRAMITE DLL

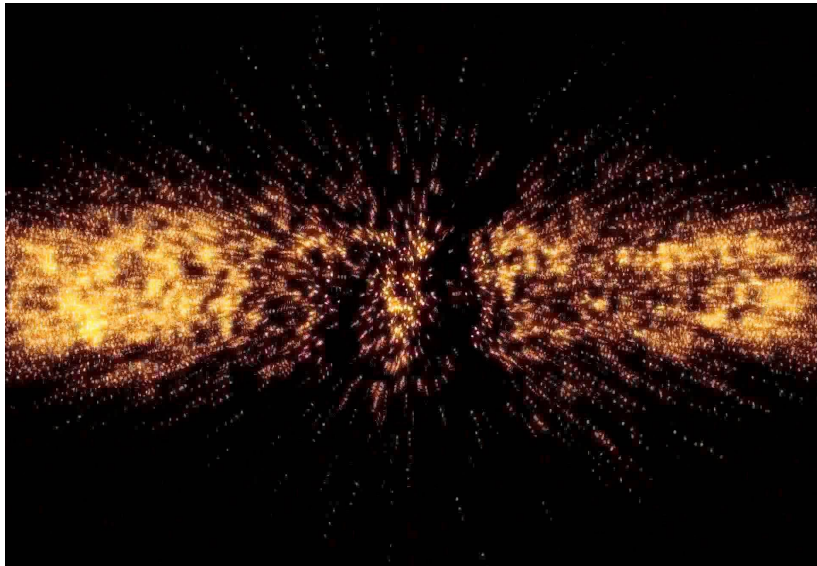
È possibile importare nuove funzioni utilizzando delle *dynamic-link library* (DLL) per estendere l'operatività di Atomic, trasformando funzioni e programmi scritti in C++ in funzioni Atomic facili da utilizzare.

```
1 #define Atomic_export extern "C" __declspec (dllexport)
2
3 Atomic_export double cubo(double n){
4     return n*n*n;
5 }
6
7 Atomic_export double somma(double a, double b) {
8     return a + b;
9 }
```

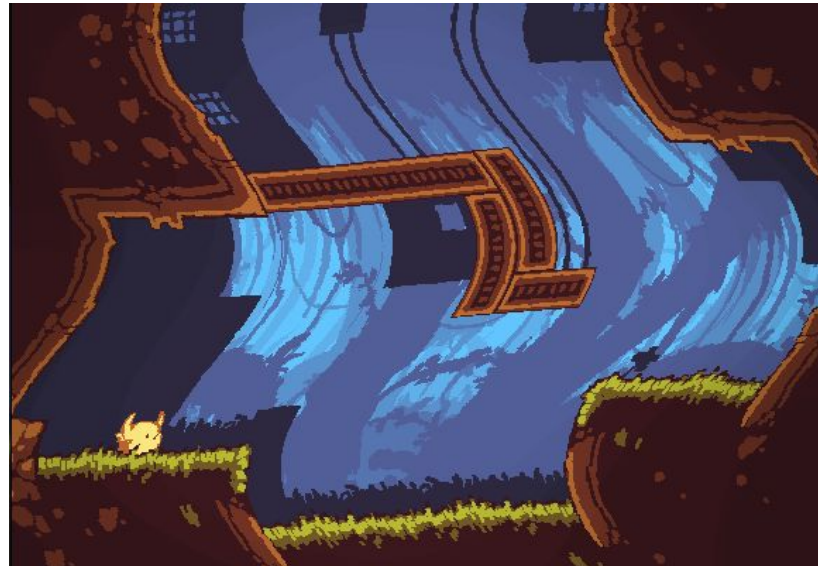


```
1 INIZIA
2 importa funzione esterna → (INDIRIZZO: "test.dll") (NOME: "cubo")
3 (TESTO: "ottieni il cubo di → (VALORE:0)")
4
5 importa funzione esterna → (INDIRIZZO: "test.dll") (NOME: "somma")
6 (TESTO: "ottieni la somma tra → (VALORE 1:0) (VALORE 2:0)")
7
8 CICLO CONTINUO
9 numero = 22
10 valore = ottieni il cubo di → (VALORE: numero)
11 disegna testo → (TESTO: "Il cubo di <numero> è <valore>")
12
13 a = 10
14 b = 3
15 valore2 = ottieni la somma tra → (VALORE 1: a) (VALORE 2: b)
16 disegna testo → (TESTO: "la somma tra <a> e <b> equivale a <valore2>") (Y: 175)
```

FEATURES FUTURE



Effetti particellari

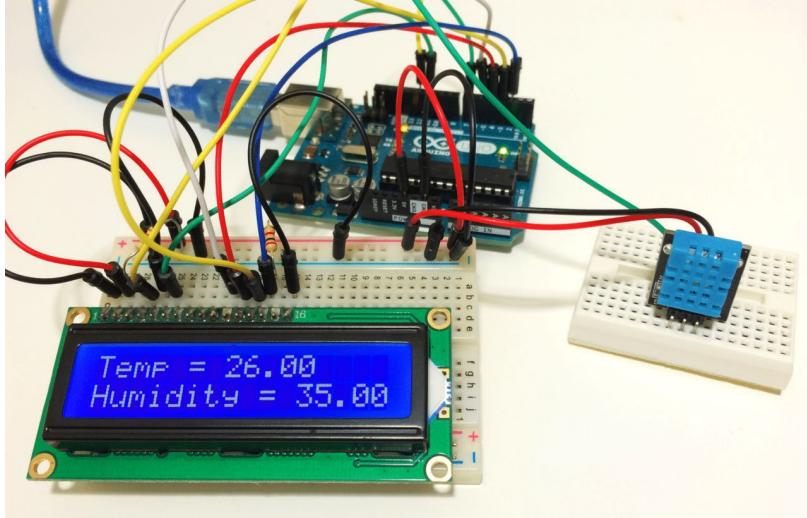


Shaders

[download](#)



FEATURES FUTURE



**Interazione
con Arduino**



**Creazione di pagine web
(preprocessing html/css)**

[download](#)



COME È STATO REALIZZATO

[download](#)



COME È STATO REALIZZATO

Atomic è stato realizzato con GameMaker Studio 2.



[download](#)



GAME MAKER STUDIO

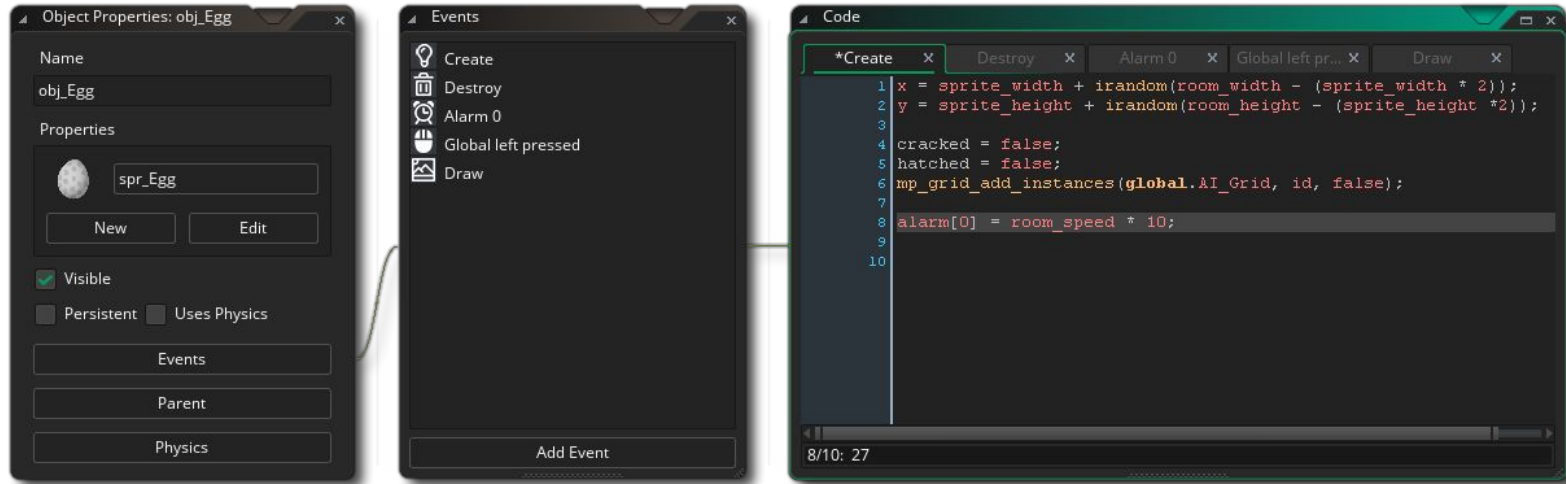
- Permette di creare videogiochi 2D (e programmi) in **tempi rapidi**
- È un progetto longevo (nato nel 1999), **consolidato** e apprezzato in tutto il mondo
- Permette di esportare per **più piattaforme** (desktop, mobile e console)
- È utilizzato nelle **scuole** e nelle università in **America e in Nord Europa**
- Il suo linguaggio di programmazione (GML) è tra i più **facili da imparare**

[download](#)



GML: GAME MAKER LANGUAGE

“GameMaker Language (GML) è il linguaggio di scripting di GameMaker. È un linguaggio imperativo, con tipizzazione dinamica, comunemente paragonato a JavaScript e linguaggi simili a C.” Fonte: Wikipedia.



COME FUNZIONA ATOMIC (IN SINTESI)

Atomic è un linguaggio interpretato.

Il codice inserito nell'editor viene letto e spezzettato in piccole parti categorizzate dette **token**.

I token vengono poi letti ed analizzati (**parsing**) per svolgere varie azioni.

I **token** subiscono poche modifiche e rimangono facilmente leggibili anche da un umano: questa pratica è molto **vantaggiosa per il debug**.

[download](#)



MULTIPIATTAFORMA

Attualmente Atomic è disponibile solo per **Windows**,

Entro la fine dell'anno saranno disponibili le versioni per **MacOS** e **Linux**.

Versioni **mobile** (Android e iOS): realizzabili ma richiedono molta ottimizzazione e un IDE ridisegnato per il mobile.

Versione **web** (html5): richiederebbe la scrittura di un convertitore **Atomic** → **javascript**

[download](#)



OPEN SOURCE

Uno degli obiettivi per il futuro è rendere il progetto il più possibile **open source**.

Sviluppare in C++ è gratis e permette l'open source.

Game Maker Studio non è gratis, tuttavia ha una versione trial e un costo accessibile (99\$ per la licenza desktop).

[download](#)



COME COLLABORARE AL PROGETTO

[download](#)



COLLABORARE COME SVILUPPATORE

- Scrivere tutorial e librerie in linguaggio **Atomic**
- Scrivere nuove funzioni in **GML**
- Scrivere DLL in **C++** per estendere le funzionalità di Atomic

Quarta opzione (in futuro): prendere parte allo sviluppo del core in GML o in C++.

[download](#)



COLLABORARE COME CREATIVO

- **Grafica:** creazione di librerie tematiche gratuite di immagini (sprite e sfondi).
- **Audio:** creazione di librerie tematiche gratuite di suoni e musiche.

Queste librerie potranno essere esterne o incluse direttamente nell'IDE.

[download](#)

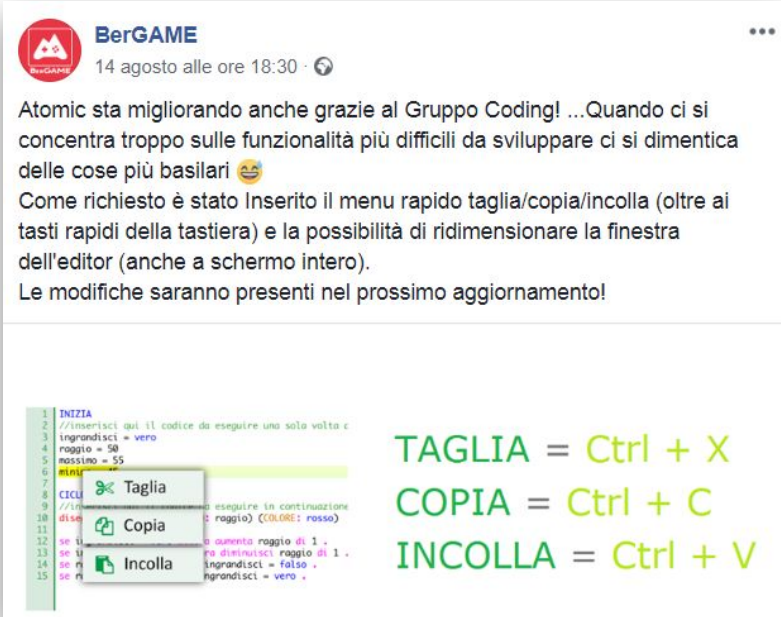


COLLABORARE COME DOCENTE

Atomic piace agli studenti ma deve anche rimanere un supporto didattico **utile ai docenti**: il **feedback** degli insegnanti è **vitale**.

Sono gli insegnanti che determinano lo **sviluppo del progetto** in base alle loro **esigenze e necessità**.

NO al coding calato dall'alto
SI al Coding migliorato dal basso



BerGAME 14 agosto alle ore 18:30 · 🌐

Atomic sta migliorando anche grazie al Gruppo Coding! ...Quando ci si concentra troppo sulle funzionalità più difficili da sviluppare ci si dimentica delle cose più basilari 😊

Come richiesto è stato inserito il menu rapido taglia/copia/incolla (oltre ai tasti rapidi della tastiera) e la possibilità di ridimensionare la finestra dell'editor (anche a schermo intero).

Le modifiche saranno presenti nel prossimo aggiornamento!

```
1 INIZIA
2 //insertici qui il codice da eseguire una sola volta e
3 ingrandisci = vero
4 raggio = 50
5 massimo = 55
6
7
8 CICLO
9 //in
10 //in
11
12 se ti
13 se ti
14 se ti
15 se ti
```

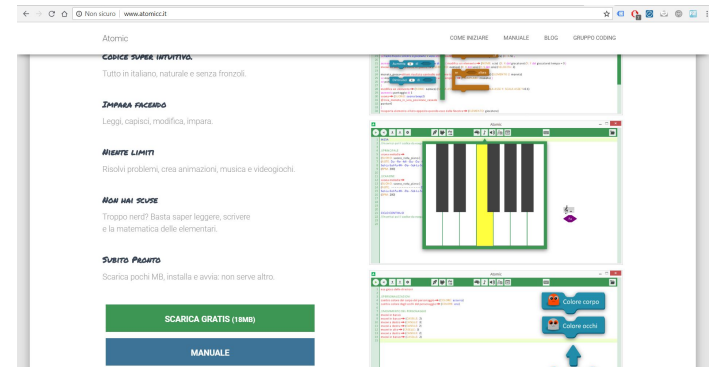
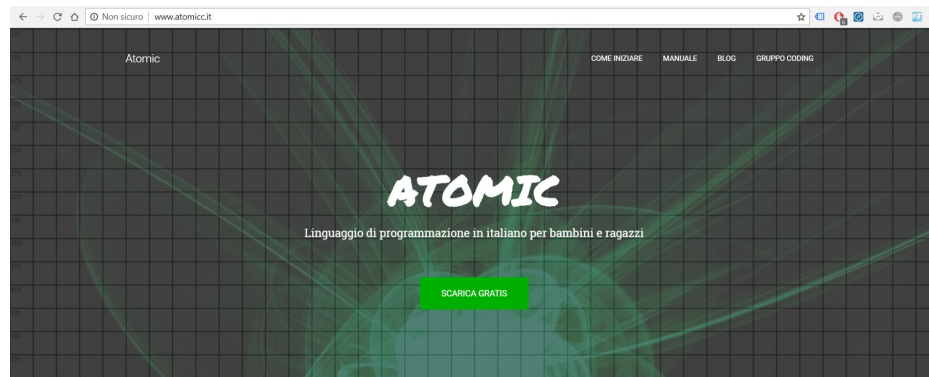
Taglia
Copia
Incolla

TAGLIA = Ctrl + X
COPIA = Ctrl + C
INCOLLA = Ctrl + V

COLLABORARE AL SITO WEB DEL PROGETTO

Il sito ufficiale del progetto è www.atomicc.it.

- Ruolo attivo **commentando** gli articoli e i tutorial, **raccontando esperienze**, dando **suggerimenti e consigli**.
- Diventare **autori** e pubblicare articoli.



COLLABORARE AI GRUPPI DI CODING



Gruppi informali che si ritrovano una volta al mese per un pomeriggio di coding con Atomic.

Senza lezioni frontali: solo una **traccia comune** da seguire ed ampliare (metodo **learning by doing**).

I gruppi di coding necessitano di **mentori volontari**.

[download](#)



CONTATTI

Per chiedere altre informazioni o collaborare al progetto potete scrivermi all'indirizzo:

francesco@bergame.eu

download



download

Thanks!



In collaborazione con

