Conway's Game of Life

C++, Python e Cython per un Automa Cellulare

Federico Pasqua

Università di Pavia

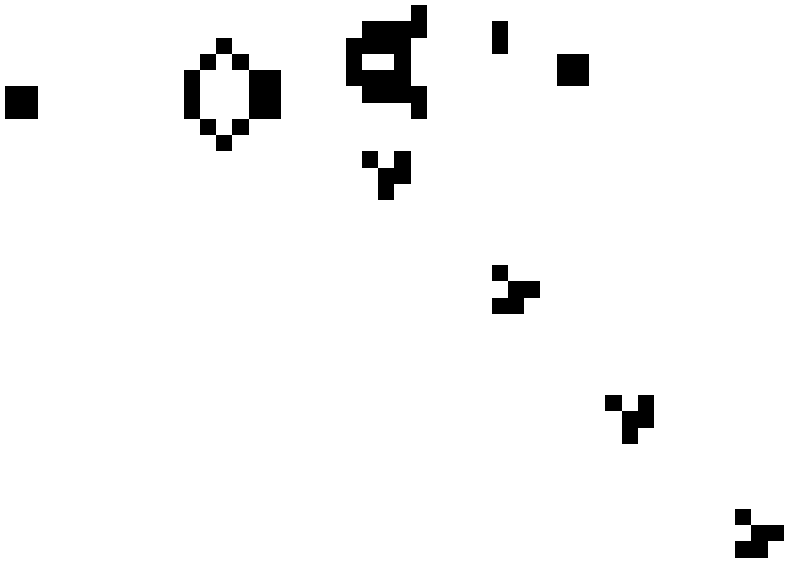
31 maggio 2018

Introduzione

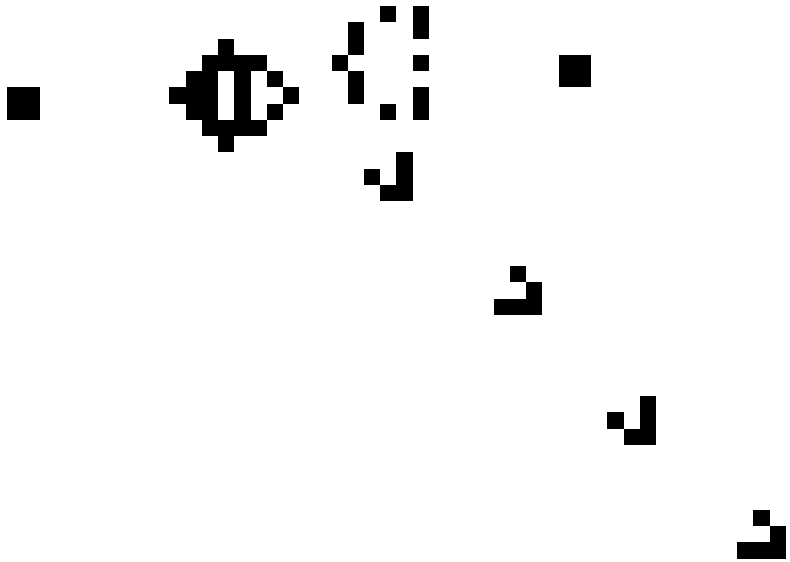
Lo scopo è creare una simulazione di un *Automa Cellulare* del tipo "Conway's Game of Life" che verrà illustrato dopo.
Inoltre verrà scritto un motore di simulazione in C++ per poi interfacciarlo ad un applicativo in Python 3 per la parte grafica e di gestione simulazione.
Per questo progetto ho ricorso a ben 3 differenti linguaggi di programmazione: C++, Python 3 e Cython per interconnettere i precedenti due.

Il Gioco della Vita(?)

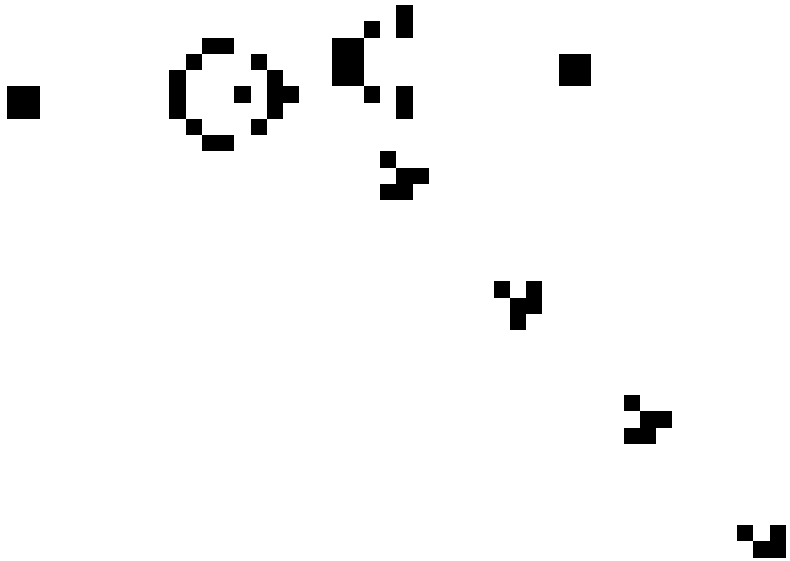
Il Gioco della Vita(?)



Il Gioco della Vita(?)

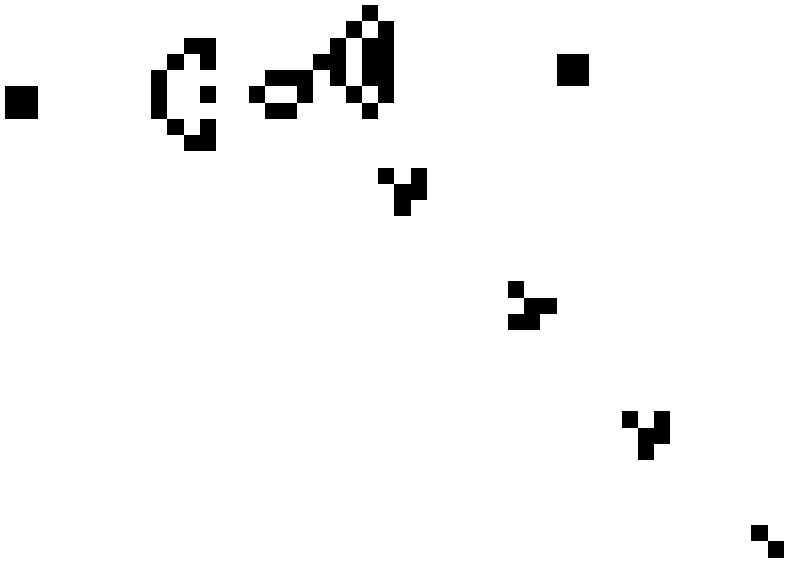


Il Gioco della Vita(?)

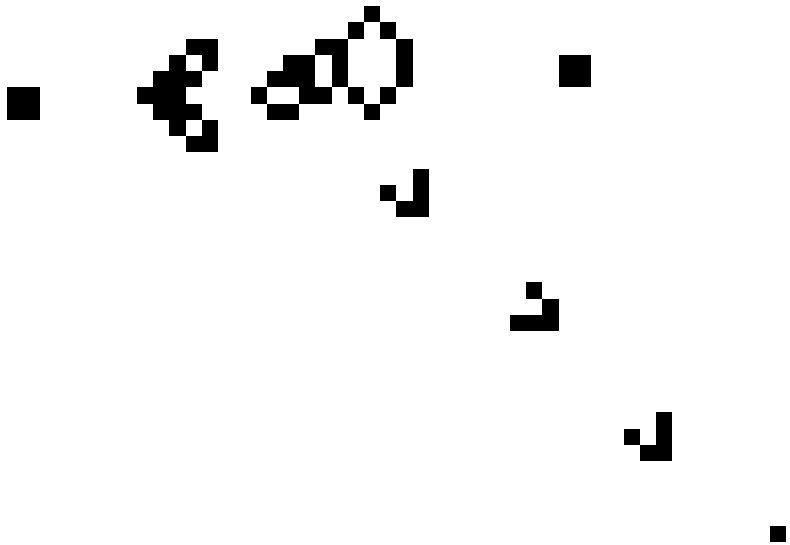


Il Gioco della Vita(?)

Il Gioco della Vita(?)

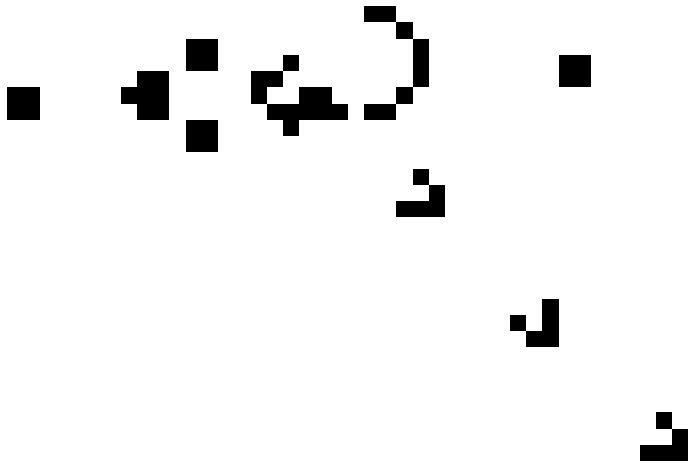


Il Gioco della Vita(?)



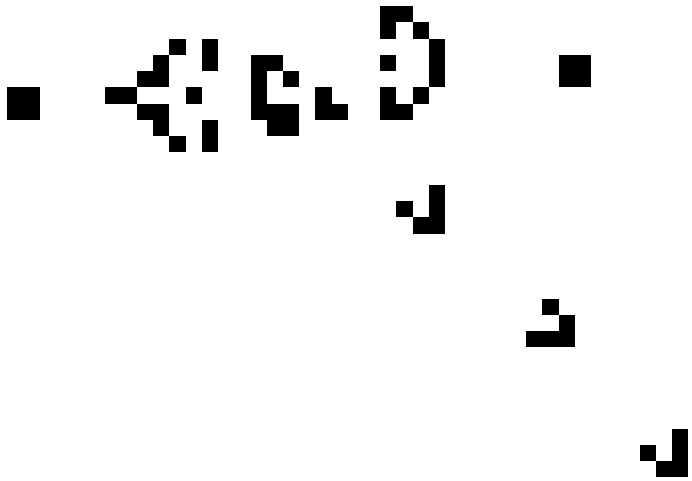
Il Gioco della Vita(?)

Il Gioco della Vita(?)



Il Gioco della Vita(?)

Il Gioco della Vita(?)

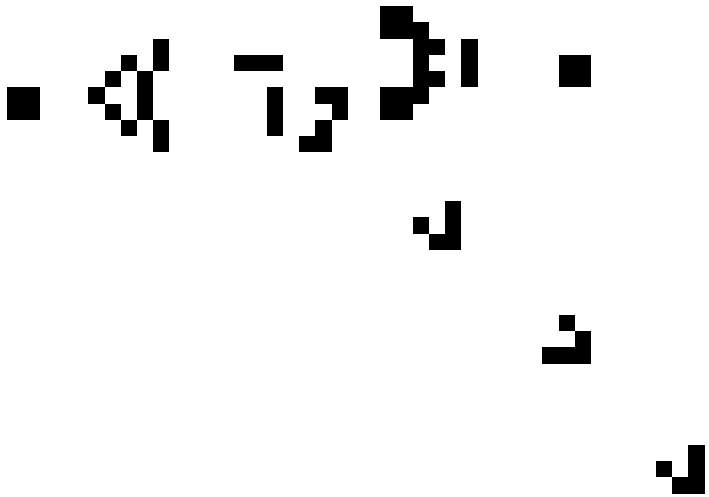


Il Gioco della Vita(?)

Il Gioco della Vita(?)

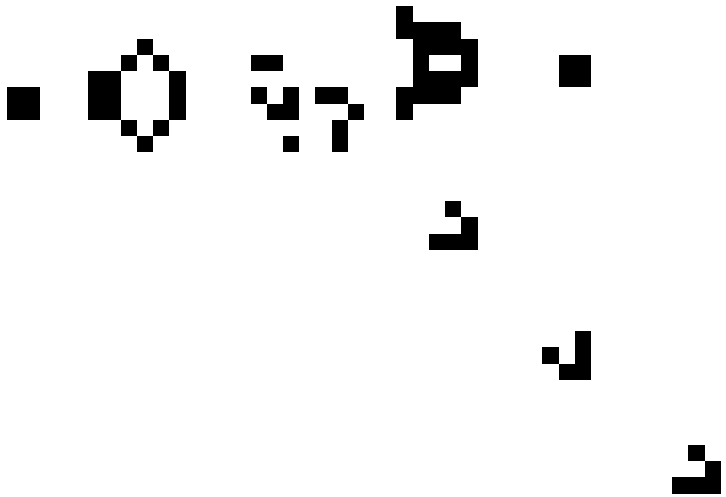
Il Gioco della Vita(?)

Il Gioco della Vita(?)

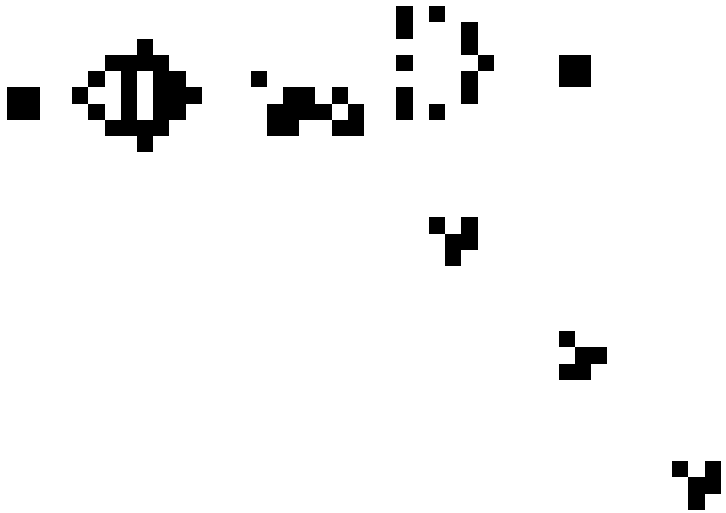


Il Gioco della Vita(?)

Il Gioco della Vita(?)

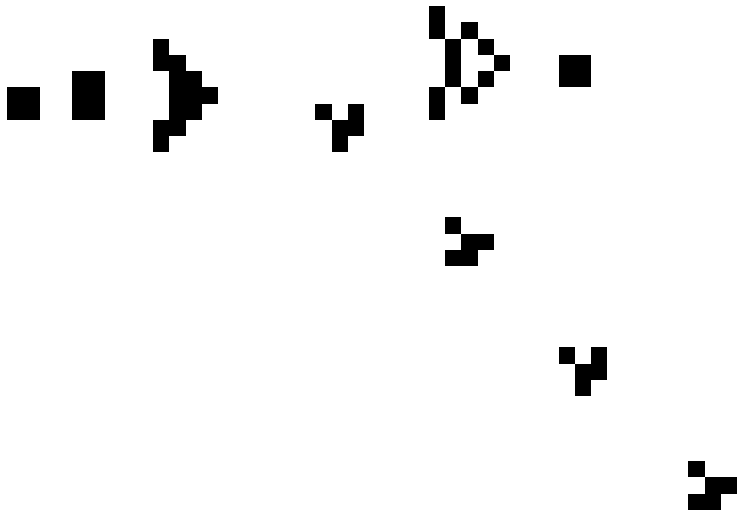


Il Gioco della Vita(?)

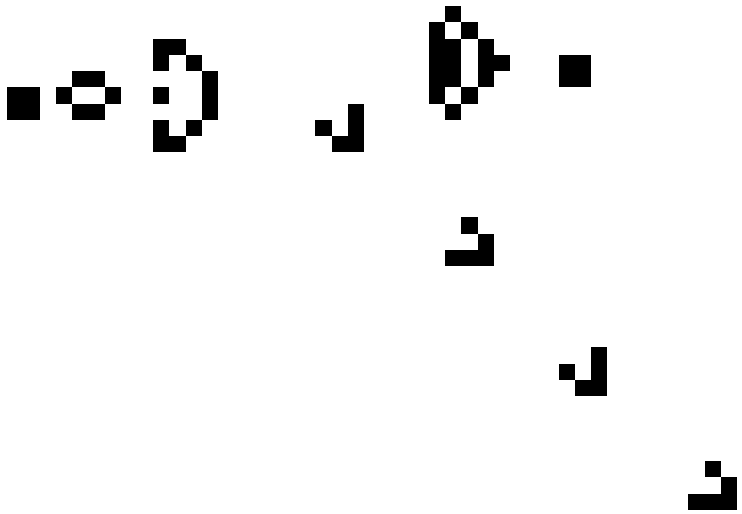


Il Gioco della Vita(?)

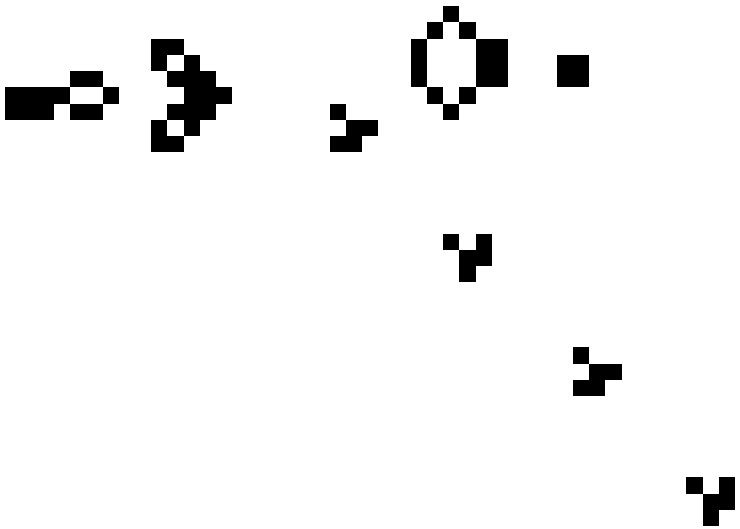
Il Gioco della Vita(?)



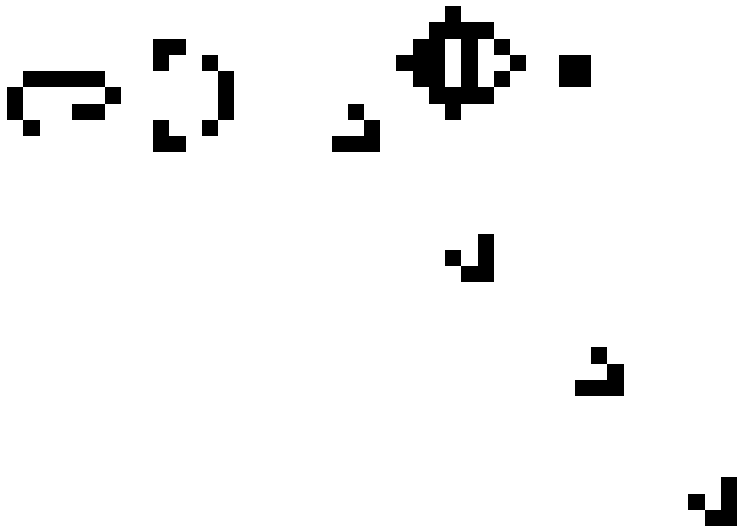
Il Gioco della Vita(?)



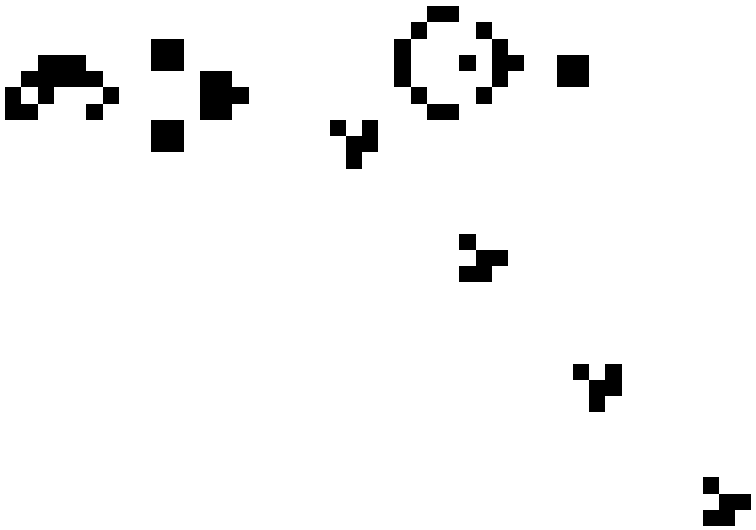
Il Gioco della Vita(?)



Il Gioco della Vita(?)



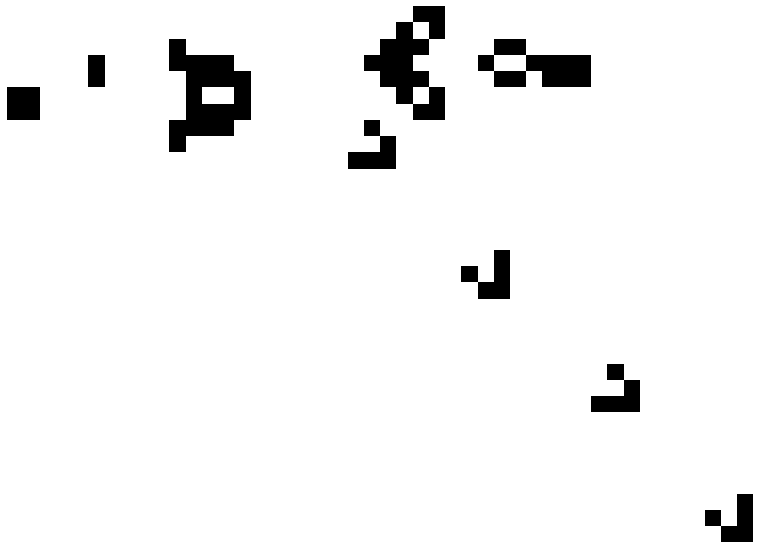
Il Gioco della Vita(?)



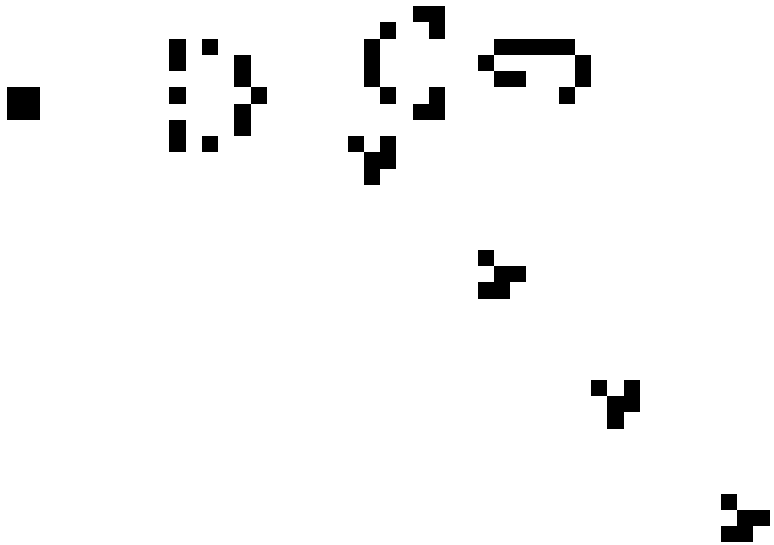
Il Gioco della Vita(?)

Il Gioco della Vita(?)

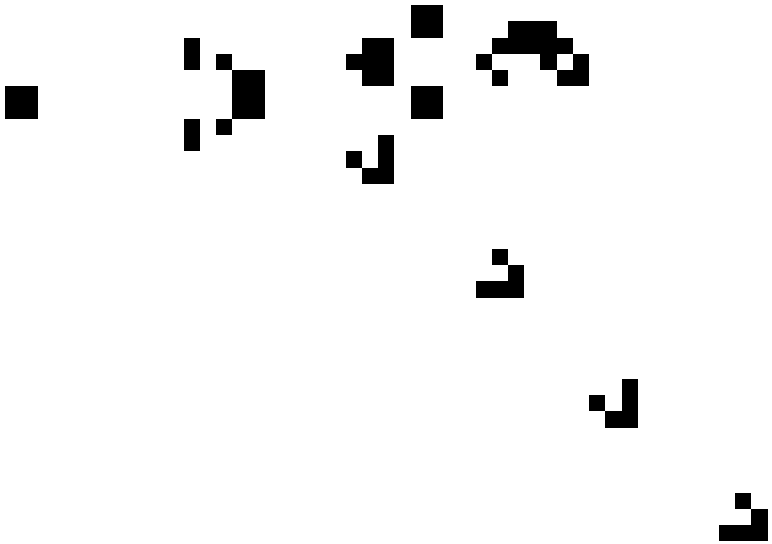
Il Gioco della Vita(?)



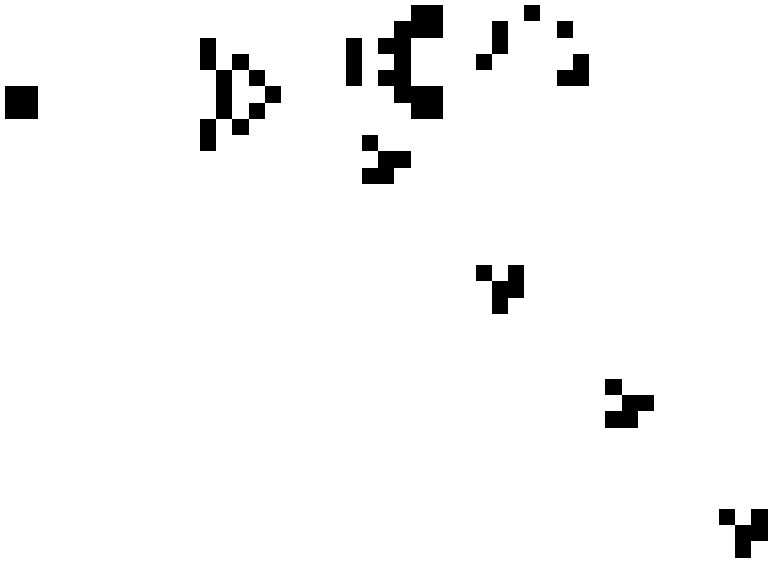
Il Gioco della Vita(?)



Il Gioco della Vita(?)



Il Gioco della Vita(?)



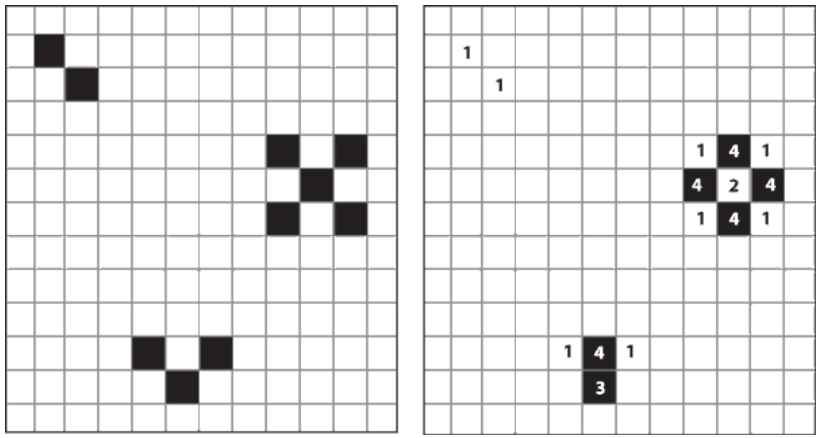
Il Gioco della Vita(?)

Il Gioco della Vita(?)

Il Gioco della Vita(?)

Una griglia infinita i cui quadrati possono assumere come valori solo 1 (vivo) o 0 (morto).
Ogni cella viene chiamata "cellula" ed è l'unità fondamentale del gioco. Il tempo è discretizzato e quindi si misura in "generazioni".
Ad ogni generazione avviene un elaborazione che porta all'evoluzione del sistema in uno stato differente.

Game of Life - The Rule



Il modo in cui avanza la simulazione da una generazione all'altra si incarna in una **Rule**. La Regola ci dice da una generazione alla successiva con quante cellule vive confinanti deve stare una cellula morta per "resuscitare" e tornare 1, oltre che quante devono stare vicino ad una cellula già viva affinché non "muoia" e torni 0.

Il Game of Life è solo questo.

La regola di Conway è detta **B3/S23**.
Ovvero che una cellula morta per tornare viva (Birth) deve avere 3 cellule vive vicino alla generazione precedente, mentre una cellula già viva per restarlo (Stay) deve avere o 2 o 3 cellule vive vicine alla generazione precedente.

Questa sarà la regola che utilizzeremo nella nostra simulazione, ma lasciando spazio ad un possibile futuro cambiamento.

Obiettivi - Riassunto

Obiettivi - Riassunto

- **Libreria di simulazione per il Game of Life**
  Questo è l'obiettivo primario. Verrà scritta in C++ per avere una buona performance e usando paradigma OOP.

Obiettivi - Riassunto

- **Libreria di simulazione per il Game of Life**
  Questo è l'obiettivo primario. Verrà scritta in C++ per avere una buona performance e usando paradigma OOP.
- **Interfaccia tra la libreria C++ e Python**
  Questo è il punto critico. Utilizzando una versione espansa del Python detta Cython, verrà creata un interfaccia Python per la libreria, senza necessità di riscriverla.

Obiettivi - Riassunto

- **Libreria di simulazione per il Game of Life**
  Questo è l'obiettivo primario. Verrà scritta in C++ per avere una buona performance e usando paradigma OOP.
- **Interfaccia tra la libreria C++ e Python**
  Questo è il punto critico. Utilizzando una versione espansa del Python detta Cython, verrà creata un interfaccia Python per la libreria, senza necessità di riscriverla.
- **Applicazione grafica interattiva**
  Questo obiettivo è il motivo per cui si è passati ad un approccio ibrido al problema, spacchettandolo e risolvendolo con ben 3 strumenti differenti. Sarà interattivo, con GUI non testuale e adattabile. Sfruttera le peculiarità del Python.

Perché il C++?

[1] Tempo impiegato per l'accesso all'esecuzione di risorse accessorie.

[2] First Release: 23 Maggio 1987; C++ Release: Dicembre 1987

Perché il C++?

- **Tipizzazione Statica**
  Ogni variabile in C++ ha un tipo ad essa assegnata. Questo riduce il tempo di esecuzione del programma riducendo di molto l'overhead[1]subito dal programma finale in esecuzione.

[1]Tempo impiegato per l'accesso all'esecuzione di risorse accessorie.
[2]First Release: 23 Maggio 1987; C++ Release: Dicembre 1987

Perché il C++?

- **Tipizzazione Statica**
  Ogni variabile in C++ ha un tipo ad essa assegnata. Questo riduce il tempo di esecuzione del programma riducendo di molto l'overhead[1]subito dal programma finale in esecuzione.
- **Linguaggio Compilato**
  Questo permette di rimuovere determinati errori già in fase di compilazione rendendo intrinsecamente più sicuro il linguaggio. Inoltre la velocità del codice finale è massima.

[1]Tempo impiegato per l'accesso all'esecuzione di risorse accessorie.
[2]First Release: 23 Maggio 1987; C++ Release: Dicembre 1987

Perché il C++?

- **Tipizzazione Statica**
  Ogni variabile in C++ ha un tipo ad essa assegnata. Questo riduce il tempo di esecuzione del programma riducendo di molto l'overhead[1]subito dal programma finale in esecuzione.
- **Linguaggio Compilato**
  Questo permette di rimuovere determinati errori già in fase di compilazione rendendo intrinsecamente più sicuro il linguaggio. Inoltre la velocità del codice finale è massima.
- **VELOCE!**
  Grazie ai precedenti due punti abbiamo infine un linguaggio che produce applicativi veloci e ottimizzati sotto ogni punto di vista, grazie anche a GCC e ai suoi 30 anni di esperienza.[2]

[1] Tempo impiegato per l'accesso all'esecuzione di risorse accessorie.
[2] First Release: 23 Maggio 1987; C++ Release: Dicembre 1987

Perché NON il C++?

[3] Boost.UI - https://github.com/kosenko/ui - pubblicata il 26/09/2017

Perché NON il C++?

- **GUI non standard**
  Non esiste nessuna libreria GUI standard, costringendo l'uso di librerie terze parti come grafica. Solo di recente la libreria `Boost` ha proposto un prototipo di standard per l'UI.[3]

[3] Boost.UI - https://github.com/kosenko/ui - pubblicata il 26/09/2017

Perché NON il C++?

- **GUI non standard**
  Non esiste nessuna libreria GUI standard, costringendo l'uso di librerie terze parti come grafica. Solo di recente la libreria `Boost` ha proposto un prototipo di standard per l'UI.[3]
- **Curva d'apprendimento ripida**
  L'impegno richiesto per impararne le funzionalità di livello medio o più è enorme, richiedendo anni anche solo ad imparare a usare un costrutto decentemente.

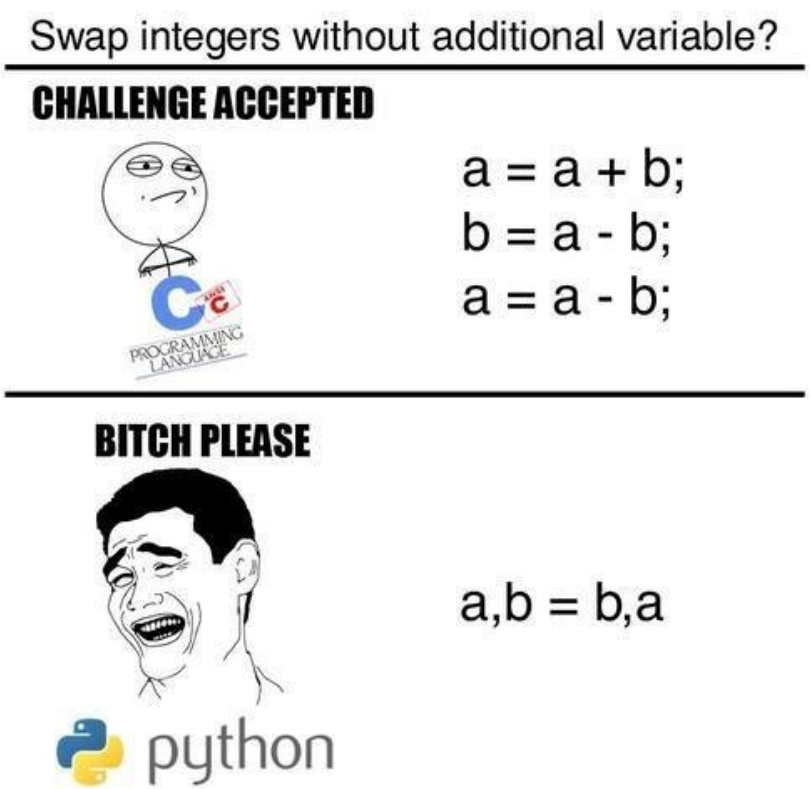
[3] Boost.UI - https://github.com/kosenko/ui - pubblicata il 26/09/2017

Perché NON il C++?

- **GUI non standard**
  Non esiste nessuna libreria GUI standard, costringendo l'uso di librerie terze parti come grafica. Solo di recente la libreria `Boost` ha proposto un prototipo di standard per l'UI.[3]
- **Curva d'apprendimento ripida**
  L'impegno richiesto per impararne le funzionalità di livello medio o più è enorme, richiedendo anni anche solo ad imparare a usare un costrutto decentemente.
- **LENTO nello sviluppo!**
  Scrivere un programma, proprio a causa della sete di specifiche del linguaggio stesso, richiede impegno e soprattutto tempo. Non puoi scrivere C++ o impararlo velocemente.

[3] Boost.UI - https://github.com/kosenko/ui - pubblicata il 26/09/2017

Cosa c'entra Python?



Il Python ha caratteristiche diametralmente opposte al C++.
Questo è dovuto anche ad un differente approccio al programma tra i due linguaggi.
Uno preferisce la stabilità e la velocità d'esecuzione, l'altro la maneggevolezza, la velocità di scrittura e la leggibilità.

Hello World - C++ vs Python

Hello World - C++ vs Python

- **C++**

```cpp
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

Hello World - C++ vs Python

- **C++**

```cpp
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

- **Python 3**

```python
print("Hello World!")
```

Perché il Python?

[4] Al 5 Novembre 2017 la repository ufficiale `PyPI` contiene 121006 moduli

Perché il Python?

- **"Questo è plagio! Usi solo librerie esterne!"**
  Python è famoso per avere un quantità di moduli incredibile[4], ed anche per l'esistenza di moduli particolarmente ottimizzati in grado di fare praticamente qualunque cosa si desideri.

[4] Al 5 Novembre 2017 la repository ufficiale `PyPI` contiene 121006 moduli

Perché il Python?

- **"Questo è plagio! Usi solo librerie esterne!"**
  Python è famoso per avere un quantità di moduli incredibile[4], ed anche per l'esistenza di moduli particolarmente ottimizzati in grado di fare praticamente qualunque cosa si desideri.
- **VELOCE nello sviluppo**
  Ottimo per scrivere programmi in poco tempo, dato che l'intero linguaggio è stato disegnato per essere utile, veloce da usare e maneggevole più di ogni altra cosa.

[4] Al 5 Novembre 2017 la repository ufficiale `PyPI` contiene 121006 moduli

Perché il Python?

- **"Questo è plagio! Usi solo librerie esterne!"**
  Python è famoso per avere un quantità di moduli incredibile[4], ed anche per l'esistenza di moduli particolarmente ottimizzati in grado di fare praticamente qualunque cosa si desideri.
- **VELOCE nello sviluppo**
  Ottimo per scrivere programmi in poco tempo, dato che l'intero linguaggio è stato disegnato per essere utile, veloce da usare e maneggevole più di ogni altra cosa.
- **I moduli di grafica sono semplici**
  Molti moduli di grafica disponibili per Python sono facili da usare o comunque molto più semplici della media di difficoltà delle librerie C++.

[4] Al 5 Novembre 2017 la repository ufficiale `PyPI` contiene 121006 moduli

Python + C++ = ?

Ma con tutte queste differenze come faccio a collegare questi due mondi così diversi?

Python + C++ = ?

Ma con tutte queste differenze come faccio a collegare questi due mondi così diversi?

Python ha una API, una libreria chiamata C-API, che permette al Python di essere interagibile da un programma in C, e quindi anche C++.
Però questa libreria è particolarmente ostica e difficile da usare.
Esiste un altra libreria appositamente pensata per il C++ ovvero la *Boost.Python*, non è però sufficientemente facile da usare per scrivere un progetto velocemente.

Cython

```python
# python_functions.py
# -----------------------
import math

def f(x):
    return math.exp(-(x ** 2))

def integrate_f(a, b, N):
    s = 0
    dx = (b - a) / N
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

```python
# cython_functions.pyx
# -----------------------
from libc cimport math

cdef double f(double x):
    return math.exp(-(x ** 2))

def integrate_f(double a, double b, int N):
    cdef double s = 0
    cdef double dx = (b - a) / N
    cdef int i
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

Cython è un superset del linguaggio Python. Esso va a compensare la più grande mancanza di Python, ovvero la velocità.
Oltre a questo è anche in grado di fornire i pro del C++ e anche di interconnettere facilmente i due environment.

Cython: Pro e Contro

Cython: Pro e Contro

Pro:

- **La sintassi di base è quella di Python.**
- **Velocità di esecuzione del C++.**
- **Compila in dynamic link libraries (DLL).** Un compilato può essere importato da qualunque istanza Python.
- **Può accedere a codice C/C++.**

Cython: Pro e Contro

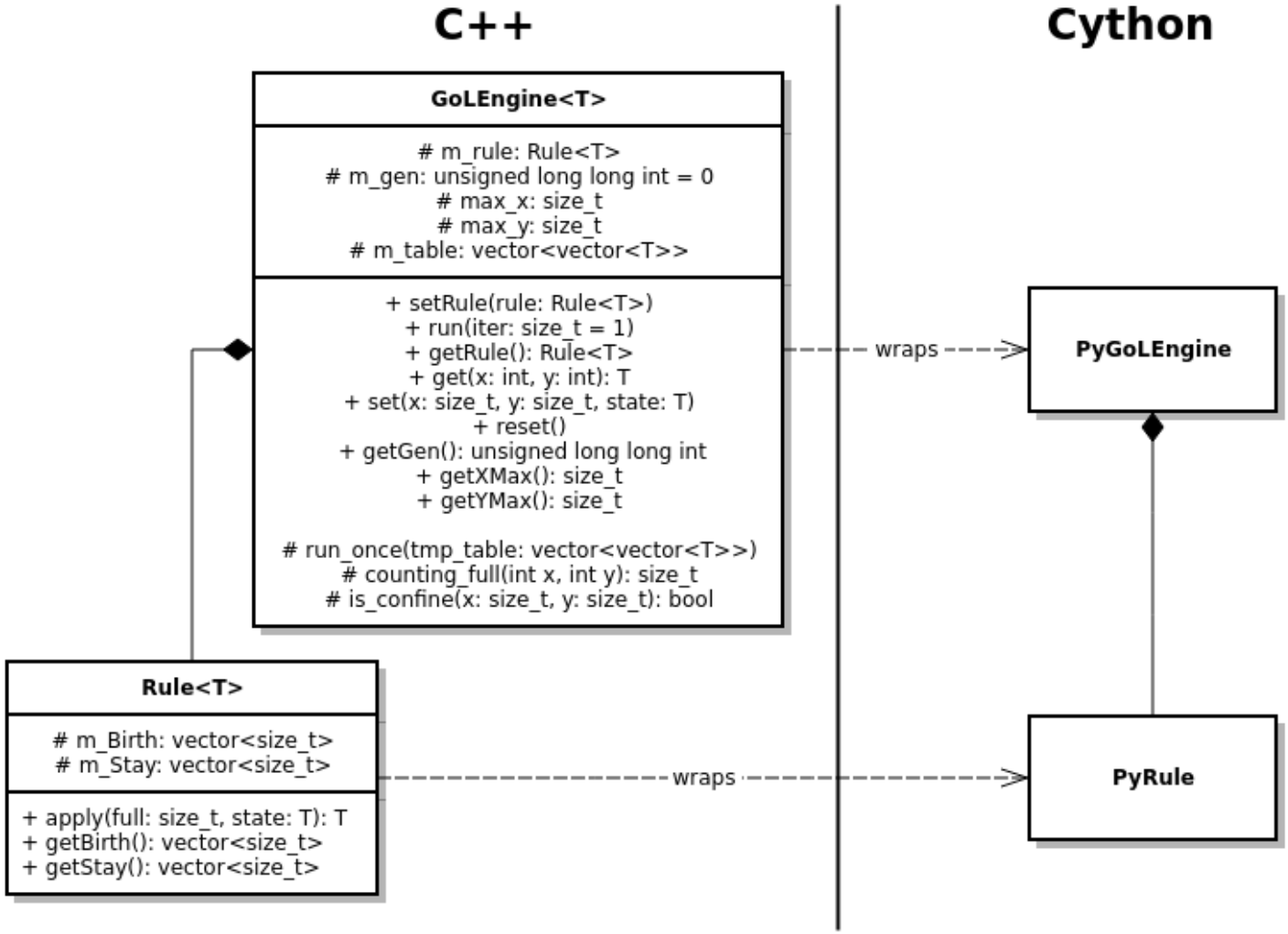
Pro:

- **La sintassi di base è quella di Python.**
- **Velocità di esecuzione del C++.**
- **Compila in dynamic link libraries (DLL).** Un compilato può essere importato da qualunque istanza Python.
- **Può accedere a codice C/C++.**

Contro:

- **Convertire librerie C++ in moduli Python non è automatico.** Viene richiesto infatti un minimo di lavoro manuale.
- **Non rimpiazza C/C++.** Cython è comunque Python. Ad esempio non potrei mai scrivere un kernel in Cython.

Architettura



Con questa divisione dei compiti mi ritrovo con tutta la parte numerica della simulazione eseguita da un codice in C++ naturalmente ottimizzato e veloce e dei wrapper pronti per l'utilizzo in Python.

Successivamente da Python verranno dati i parametri della simulazione.

Fare l'intera simulazione in C++ avrebbe dato una velocità imbattibile, ma il tempo necessario alla sua realizzazione sarebbe stato eccessivo visto anche l'obiettivo di poter facilmente interfacciare una GUI alla simulazione

Tour del Codice

Ora illustrerò il codice.

Non utilizzerò le slide data la difficoltà di condensare *"le cose giuste"* nelle sole slide.

Esempi Python + C++ nel mondo reale

Esempi Python + C++ nel mondo reale

- **SciPy ecosystem** Ecosistema di moduli Python (`NumPy`, `SciPy`, `MatPlotLib`, `Pandas` e `SymPy`) per il calcolo numerico, scientifico, simbolico e la manipolazione e visualizzazione di dati.

Esempi Python + C++ nel mondo reale

- **SciPy ecosystem** Ecosistema di moduli Python (`NumPy`, `SciPy`, `MatPlotLib`, `Pandas` e `SymPy`) per il calcolo numerico, scientifico, simbolico e la manipolazione e visualizzazione di dati.
- **Scikit-learn** Modulo di Machine Learning per Python. Le parti che avevano bisogno di elevata velocità sono state implementate in Cython e C++.

Esempi Python + C++ nel mondo reale

- **SciPy ecosystem** Ecosistema di moduli Python (`NumPy`, `SciPy`, `MatPlotLib`, `Pandas` e `SymPy`) per il calcolo numerico, scientifico, simbolico e la manipolazione e visualizzazione di dati.
- **Scikit-learn** Modulo di Machine Learning per Python. Le parti che avevano bisogno di elevata velocità sono state implementate in Cython e C++.
- **Quora** Piattaforma di domande e risposte con un certo livello di contenuto. Scritto interamente in Python, ha portato i bottleneck in Cython.

Esempi Python + C++ nel mondo reale

- **SciPy ecosystem** Ecosistema di moduli Python (`NumPy`, `SciPy`, `MatPlotLib`, `Pandas` e `SymPy`) per il calcolo numerico, scientifico, simbolico e la manipolazione e visualizzazione di dati.
- **Scikit-learn** Modulo di Machine Learning per Python. Le parti che avevano bisogno di elevata velocità sono state implementate in Cython e C++.
- **Quora** Piattaforma di domande e risposte con un certo livello di contenuto. Scritto interamente in Python, ha portato i bottleneck in Cython.
- **Tensorflow** Enorme libreria di machine learning e reti neurali mantenuta da Google. Scritta in C++ e wrappata avendo come bersaglio Python. Sono disponibili anche API per altri linguaggi quali appunto C++, Java e Go.

Grazie dell'attenzione