SIMULAZIONE DI UN'EPIDEMIA

Lorenzo Califano, Martino Garelli, Martino Pasqualotto, Francesco Vassallo

INTRODUZIONE

Il progetto ha come obiettivo la simulazione di un'epidemia. Lo fa attraverso due approcci. Nel primo modo si implementa un modello matematico basato su 3 equazioni differenziali e che considera la popolazione divisa in tre gruppi, il modello SIR. Nel secondo modo si usa un modello cellulare, un insieme di circonferenze vagano in uno spazio chiuso rappresentanti individui che fanno progredire l'epidemia.

ISTRUZIONI PER ESEGUIRE IL PROGRAMMA

È necessario avere nella cartella i file *"epidemic.hpp", "epidemic.cpp", "isto.hpp", "isto_sfml.hpp", "isto_sfml.cpp", "circles.hpp"*, *"circles.cpp", "arial.ttf", "comment_font.ttf", "data_font.ttf"*, "*main.cpp".*

Per compilare il programma si può procedere dal terminale con il comando

$ g++ -Wall -Wextra -fsanitize=address -lsfml-system -lsfml-window -lsfml-graphics epidemic.cpp isto_sfml.cpp circles.cpp main.cpp

È necessario avere installata la libreria grafica "SFML".

Si ricorda che l'eseguibile deve essere aperto da terminale e con un xServer disponibile nel caso l'esecuzione avvenga su una macchina virtuale.

MAIN

Il programma principale è diviso in due parti, il modello SIR e il modello cellulare. Appena avviato verrà chiesto all'utente di scegliere una delle due parti. Essendo state sviluppate indipendentemente verranno spiegate in seguito separatamente, ricordando però che l'input/output e le funzionalità fanno parte dello stesso file *main.cpp*.

MODELLO SIR

INTERFACCIA UTENTE (MODELLO SIR)

Il modello SIR permette di calcolare l'evoluzione nel tempo di tre gruppi: infetti, suscettibili e rimossi. L'evoluzione è matematicamente derivata da due parametri β e γ, che rappresentano rispettivamente la probabilità di contagio di un suscettibile e la probabilità di guarigione (o morte) di un infetto. L'utente può scegliere se visualizzare l'andamento dei tre gruppi tramite una raffigurazione numerica stampata a schermo, o tramite istogrammi.

PRINCIPALI SCELTE PROGETTUALI (MODELLO SIR)

La prima parte del main è basata su tre header_file *("epidemic.hpp", "isto.hpp"* e *isto_sfml.hpp) e due* file.cpp (*"epidemic.cpp", "isto_sfml.cpp").*

Si crea prima di tutto la *struct "popolazione"* che contiene gli elementi utili al modello epidemico (Suscettibili, Infetti e Rimossi indicati con le rispettive iniziali maiuscole).

La classe che controlla l'epidemia è *"epidemic"*, contiene i dati di partenza dell'epidemia come membri privati. I primi controlli sui dati che verranno presi in input sono già nel costruttore nel quale viene inoltre utilizzato l'algoritmo *"round"*, il quale verrà usato spesso per poter lavorare con degli interi arrotondati correttamente. La funzione membro centrale è "*evolve*", nel quale sono

implementate le equazioni di questo modello. Si è scelto di lavorare con dei valori double, però le persone sono unità finite quindi si è creata la funzione "*approx"* per ogni qualvolta sia necessario stampare o lavorare con una popolazione intera.
Per la rappresentazione con istogrammi dell'epidemia si sono create due classi diverse. La prima è la classe *isto,* è stata pensata per essere un contenitore di dati e la seconda classe, *Finestra,* si basa su di essa. La classe *isto,* ha solo un membro privato, un *vector*, e, per farla più generale possibile, è stata scritta usando i template. Ha diverse funzioni, quelle che *Finestra* usa sono la funzione *add*, per aggiungere elementi al vettore privato, la funzione *get,* per accedere alla parte privata della classe e la funzione *max,* che semplicemente ritorna il massimo tra gli elementi del vettore.
Lo scopo della classe *Finestra* è immagazzinare dati e stamparli a schermo sotto forma di istogramma. Per tutta la parte grafica contenuta nella classe si è fatto uso di una libreria esterna chiamata SFML. Ogni dato è rappresentato da un rettangolo la cui altezza varia in base al massimo attuale, e si dispongono uno accanto all'altro. La classe stampa automaticamente anche i due assi con delle informazioni riguardo i dati.

INPUT/OUTPUT (MODELLO SIR)
Una volta selezionato dal menu principale il programma inizierà a chiedere all'utente i dati relativi all'epidemia che si desidera simulare: nell'ordine bisognerà inserire popolazione totale, numero degli individui infetti di tale popolazione e il valore di β e γ (entrambi strettamente compresi tra 0 e 1); si aprirà allora un secondo menu da cui è possibile selezionare il tipo di visualizzazione che si desidera. Sono disponibili una visualizzazione esclusivamente numerica, che stampa a schermo i valori dei tre gruppi dell'epidemia per il numero di giorni selezionato (o finché l'epidemia non si ferma), oppure una visualizzazione grafica che aprirà una finestra in cui si genererà un istogramma relativo al gruppo desiderato.

STRATEGIE DI TESTING (MODELLO SIR)
Per tutte e 3 le classi si sono effettuati dei test usando doctest (*epidemic.test.cpp, isto.test.cpp* e *isto_sfml.test.cpp*), il controllo è stato per la maggior parte sul corretto funzionamento delle singole funzioni membro. Nel caso della classe *epidemic* c'è una categoria di valori che non possono essere accettati, come per esempio i parametri al di fuori del loro dominio (essendo probabilità devono rimanere compresi tra 0 e 1) oppure la popolazione non può essere negativa. Per evitare di costruire oggetti con valori proibiti si sono messi una serie di if nel costruttore della classe, al cui interno (ossia se arriva un valore proibito) sono presenti dei *throw runtime_error*. Sempre per la classe *epidemic* sono stati messi diversi *assert* per controllare che il numero di persone rimanga costante nel tempo.

MODELLO CELLULARE

INTERFACCIA UTENTE (MODELLO CELLULARE)

Il modello cellulare consiste in una simulazione di un'epidemia, in cui le persone sono rappresentate da cerchi che si muovono di moto randomico in una finestra chiusa bidimensionale. Ogni persona cambia colore a seconda del suo stato di suscettibile (blu), infetta (rossa), guarita (verde) e morta (nera).

Ogni persona suscettibile che si trova a una distanza minore di *infection_distance* con un'altra persona infetta, può infettarsi secondo una certa probabilità. Inoltre, ogni persona infetta può, secondo leggi randomiche, morire o guarire; le persone guarite possono a loro volta tornare suscettibili, sempre secondo leggi randomiche.

PRINCIPALI SCELTE PROGETTUALI (MODELLO CELLULARE)

La seconda parte del main è basata su un header file, *"circles.hpp"*, e un file cpp, *"circles.cpp".*

Ogni persona è implementata attraverso la classe P*erson,* definita in circles.hpp, che contiene gli attributi di posizione, velocità, stato di infezione e metodi che permettono di modificare o accedere agli attributi. Nel main viene creato un vettore di persone, chiamato *people*, e viene avviato un loop, in cui ogni *Person* nel vettore *people* continua a muoversi a schermo, infettare o essere infettato. Da notare la funzione *evolve_v()* che modifica la velocità di ogni pallina secondo leggi randomiche che tendono con maggior probabilità a decrementarla rispetto che a incrementarla, così da dare un miglior effetto visivo di spostamento.

INPUT/OUTPUT (MODELLO CELLULARE)
Una volta aver scelto "Automi cellulari", il programma chiede in input il numero di persone totali, che, al fine di mantenere una buona fluidità, viene accettato solo se inferiore a 350. Nonostante ci sia questo limite massimo, si consiglia di inserire un numero di persone intorno a 250, così da capire visivamente al meglio l'evoluzione della epidemia.
Il programma chiede successivamente il numero di infetti iniziali, segue allora l'apertura della finestra grafica dove viene mostrato l'evoluzione della epidemia.
È possibile interagire con l'evoluzione cliccando con il tasto sinistro e destro del mouse: nel primo caso vengono aggiunte in quel punto persone suscettibili, nel secondo persone infette.
Alla fine dell'epidemia, se il virus viene debellato, appare una schermata che recita "Humanity defeated coronavirus", in caso contrario, ovvero se il virus riesce a uccidere tutte le persone, appare una schermata che recita "Coronavirus defated humanity".

STRATEGIE DI TESTING (MODELLO CELLULARE)
I test sono all'interno del file *circles.test.cpp* e fanno uso dello strumento "Doctest".
Essi testano il corretto funzionamento delle principali funzioni membro all'interno del programma.
Non è stato però possibile eseguire test sull'evoluzione dell'epidemia a causa delle leggi randomiche che la governano.
Inoltre, al fine di garantire il corretto funzionamento, sono stati inseriti *assert* e *throw expression.*

NOTA AGGIUNTIVA
È possibile che alla chiusura del programma vengano visualizzati dei messaggi di *memory leaking*, tuttavia essi non hanno nulla a che vedere con il programma in questioni, ma sono da ricondurre a possibili problemi interni di SFML. Essi appaiono infatti anche con programmi esempio presenti nel sito ufficiale di SFML. Per coordinarci meglio, abbiamo fatto uso di Git e Github, la repository è pubblica e a questo indirizzo: https://github.com/vass-f/progetto_pf2022.