Project CS100 S2 / /

28100208 Reem Waheed
28100055 Muhammad.

TA Advisor: Syed Mustafa Hasan

Forest Fire Simulator:

-> inspired by Conway's Game of Life

Rules:

-> 2D grid (array) representing different states of the fire

0 -> empty cell (black)
1 -> Tree (green)
2 -> Fire (red)
-1 -> Fire extinguishing zone (blue)

Rules for grid update

-> Tree next to fire -> fire (1) later changed to pink
-> Fire -> empty
-> Empty -> unchanged
-> Extinguishing zone -> Doesn't let fire spread

Edge Cases to Test:

① cells on the boder of the grid
② checks that fire burning is controlled
③ ensure that empty cells are handled properly

Use of the GUI

-> We used SFML

Installation Process:

* Followed a tutorial on Github Labs
* Downloaded Msys 64 compiler from Msys2
* Run commands in Msys2 terminal to install toolchain and SFML library
* Go to edit environment variables and select path: copy path of Msys64
* Write commands in the Windows powershell

Creating SFML

* Make folders
* Structure folders in terminal (commands)
* Copy paste required code
  - main
  - build
  - sfml - works
  - gitignore

* Write commands into G++ / /

* Installation Complete *

→ Novel feature:

introduced file handling
→ save the current array state in a file
→ load it from the file

→ progress is saved

* Flow of logic of code

Structure

→ initialize a 2D array with dimensions 20×20 (seems sufficient)
→ give each "cell" in the array a size of 30 pixels for the SFML
→ create another array, temp, to navigate updates

Propagation

→ Randomly populate grid (70% trees, 30% empty)
→ Randomly place fires

File handling

→ Open a file to write dimensions of the array
→ Then write the values in it, and let the user know where this is saved

-> Open mat file and read the state / data, then close the file

Rendering
-> Create a rectangle
-> Assign colors to array "cells"

New Generation
-> Update the grid according to the rules, and use temp grid
-> Then copy temp grid to grid

Main function
-> call render function
-> Initialize grid
-> Flag to control running of simulation
-> Events: user input, controls opening/closing the window of the simulation
-> save / load the grids
-> When the window is cleared, the grid is drawn and displayed.

User Input
Space bar: start / pause
S: save the grid
L: load the grid
Mouse click: Place zones

↳ return to most recently saved grid

and read me
men close

angle
o array "cells"

eneration
d according
use temp grid
p grid to

hon
nion

nning of simulation
, controls
window of

'ds
is decred,
and

ut
ause
rid
id
2 zones

ently

Need for Temp

-> A way to prevent errors
-> So that changes are only applied to the grid after the generation proceeds

Testing Edge cases

-> check if file is opening (load grid) and display a standard error
-> if the inFile doesn't give the correct dimensions (that match the grid) an error is displayed
-> If the data in the file doesnot match the data in the grid, display an error

-> Grid only updates in the end to prevent any errors

Clock function =

->make sure that the grid updates in real time.
-> restart the clock each time the cycle re happens

-> condition :check if 500 milliseconds have passed. If true, update the grid and reset the clock

Role of a task.json file

-> Transports data from one to the other
-> We don't have to manually update in the terminal, the task.json file does that for us
   ↳ builds the object file from the cpp file and then uses the object file to build the exe file

What is an Object

Window -> canvas

attributes -> size / color / title
methods
window { ↳ draw shape
         ↳ pollevent

-> Classes are used due to SFML
   ↳ groups ~~data~~ functions (methods) and attributes together

a task-json file

one to me

manually update
task.json file

ject file from
men uses me
uild me exe

Object

butes → size/ color/ title
ods
↳ draw shape
↳ pollevent

e to

ns and
ds)

attributes
together

Classes used:

* RenderWindow → obj is window
* Event → event (user interactions)
* RectangleShape → cell shape

Adding code complexity

→ Fire randomly extinguishes (20% chance)

→ Fire first turns yellow, to symbolize it losing intensity

→ Keyboard r → respawn grid (restart)

→ left click → ~~extinguisher~~ fire
right click → extinguishing

→ extinguishing zone change → to pink

→ Adding music to me simulation :

when more trees → peaceful
when more fire → sinister

↳ use sf::Music
count trees and fire

→ compile extra

getstatus : function / built in
retrieves current playback
state

→ Added a splash screen

→ Imported font
Positioned it on grid

→ Added PNG file

Texture : Makes object
Sprite : Displays