CS 211 Lab 7

Bejeweled

Winter 2021

We will be looking at yet another a C++ program using the GE211 game engine in a reasonably advanced example game. You may have played this game in various forms such as in Candy Crush or other tile swapping games, but the basic concept is to destroy sets of like tiles by swapping two tiles to create a set. In this version of the game a set will be considered 4 or more tiles of the same type.

This game uses the model–view–controller pattern not-yet-described in class, which allows defining the look, user interaction, and "business logic" of an interactive program as separate components. Provided are the `Model` class which defines the internal game state, the `View` class for rendering game to the screen, and the `Controller` class for reacting to user input and tying it all together. In addition, because the tiles in this game are rather complex themselves, we have tile.hxx which sets up `Board::position`, an `Tile::apply_action` function to apply an action which is presumed to be a subclass of `Tile::Action` to the tile. (a different subclass of `Tile::Action` can provide different actions for special types of tiles), a `Tile::symbol` to represent the text inside the tile (for special types in this case. We can also use another representation strings for normal tiles). All this allows creation of more tile handlers which can each have special destructive powers. You can see that we have provided the normal action handler (which returns an empty set, meaning that for the specific action, we just want to follow the normal Bejeweled rules), and a horizontal lazer (deletes all tiles in the row in addition to the set we created by swapping), inside actions.cxx.

Lab setup

Project setup

For C++ projects, including this lab, the starter code is provided as a ZIP file for you to download: https://nu-cs211.github.io/cs211-files/lab/lab07.zip. Extract the archive file into a directory in the location of your choosing. Once you have your new directory containing the starter files, you can open it in CLion.

Be careful, as CLion will only work correctly if you open the *main project directory* (which has the the CMakeLists.txt in it). If you open any other directory, CLion will create a CMakeLists.txt for you, but it won't work properly.

General idea

The version of Bejeweled that you have been given a board inside bejeweled.cxx (defaults to 10 by 8) with several (defaults to 6) groups - which will behave as tile colors for grouping same colors - and as many types as you have tile handlers (starting from 2). From here, the controller decides when to update a frame and in each update `model_.step()` is called which is the brains behind finding what needs to change, detecting the set of tiles to be destroyed (including any caused by destroying a special type), and removes them as needed. Looking through this function and the other functions it uses in the `Model` class should help you understand how the game is utilizing tiles. The `Controller` also utilizes some of the `Model` functions in `Controller::on_mouse_up` ,which (when the view isn't going through animations) on first click of a valid tile will select that tile and on second click of another tile will attempt to swap them. Upon swapping and creating a set to be destroyed, that set of tiles will be removed and the tiles above them will be shifted down (`Model::falling_step`), and new random tiles will also be shifted down to fill the gaps created at the top. All of these changes are animated by the `Controller` class (which will make the program unresponsive to input while it displays the changes slow enough for you to actually see what happens. See the implementation of `Controller::can_animate()`).

More Valid Swaps

Let's put more excitement in our game by adding new features.

1. Change actions.cxx and the main class (bejeweled.cxx); and add a new tile with special new features. For the lab, you can just follow the logic behind adding horizontal lazer type of action.
2. Add a sprite at the top left corner to keep the score. Refer to the previous lab to figure out how.