

Version: 1.1  
Author: Ivan Vedrau  
Date: November 2025

## The Four-Color Theorem using SDL2

### Program goal

The purpose of this program is to visualize and demonstrate the Four Colour Theorem through an interactive game. It is designed not only for entertainment but also as an educational tool to help understand the theorem in an intuitive way.

The theorem states that any map divided into contiguous regions can be coloured using no more than four colours so that no two adjacent regions share the same colour.

The user's goal is to correctly colour all regions of a randomly generated map using only four colours, following the rule above.

The program highlights any conflicts (adjacent regions having the same colour), measures the user's completion time, and maintains a Hall of Fame list with the best completion times.

### Program behaviour

When started, the program opens a graphical window displaying:

- A randomly generated map of connected regions.
- A colour palette with four selectable colours.
- An information bar (HUD) showing the current colour.

The player has to select any of the 4 colours and click regions to fill them. The program prevents no action but visually highlights conflicts (regions that violate the four-colour rule). The number of mistakes will be stated in a hall of fame after the game.

When all regions are correctly coloured without any conflicts, the program displays a "Victory" screen, records the time, and allows the player to enter their name for the Hall of Fame.

The player can adjust difficulty, restart the game, or view the Hall of Fame at any time.

### User interface and controls

The main window is divided into two parts:

*Map area: top section (majority of the screen)*

*HUD area: bottom section (about 100 pixels high)*

- Palette of four colours (clickable squares).
- Current colour indicator.
- Number of regions (difficulty).
- Control shortcuts (R, ESC).

Program works through inputs with a mouse and a keyboard. Mouse inputs are so far as follows:

- **Left click on map** Colour the selected region with the currently selected colour.
- **Right click on map** Remove colour from the selected region (reset to uncoloured).
- **Left click on palette** Change the current selected colour.

Keyboard inputs:

- **1-4** Select colour #1–4 directly.(RGB and yellow)
- **R** Regenerate the map with the same difficulty (new random layout).
- **ESC** Exit the program

## Program states/Program Flow

**Main Game** Default state. User can colour regions and see updates in real-time.

**Victory / Input Name** Appears when the user correctly colours the map. A popup asks for a name (keyboard input) to record in Hall of Fame.

**Hall of Fame** Shows top completion times and corresponding player names.

Start → Random map generation

User colours regions → conflict detection

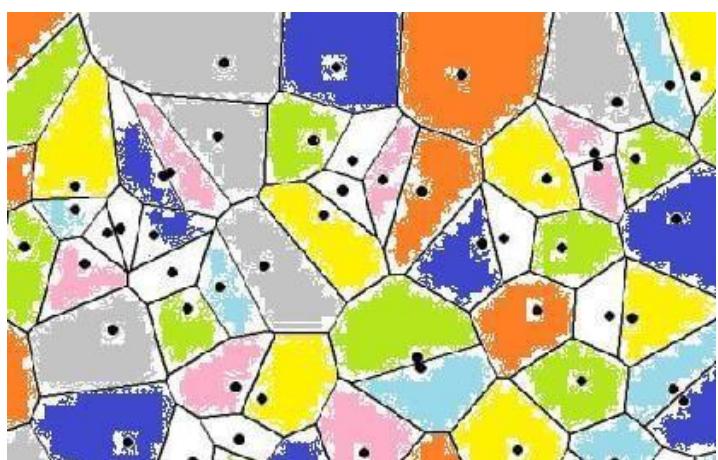
All valid → Victory screen → save score

Return to game or exit

## Map Generation

Each new game creates a random map made of contiguous regions using a Voronoi-like algorithm: A chosen number of random “seed” points (depending on difficulty) are distributed across the screen.

- Every pixel is assigned to its nearest seed, forming irregular contiguous regions.
- A matrix of adjacency relationships is computed — defining which regions touch each other.
- Difficulty determines the number of regions:



### *Approximate number of regions*

|        |       |
|--------|-------|
| Easy   | 10–20 |
| Normal | 30–70 |
| Hard   | 70–80 |

### **Difficulty**

The valid range is 10–80 regions, but the default value is 24 regions. To prevent errors values outside this range are automatically clamped to the nearest valid value.

Pressing 'R' generates a new random map using the current difficulty.

### **Colouring logic and conflict detection**

Each region can be in one of five states:

uncoloured, colour1, colour2, colour3, colour4.

After each colouring action:

The program checks all adjacent pairs of regions. If any two adjacent regions have the same colour, both are marked as conflicting. Conflicting regions are drawn in a brighter shade of their colour to alert the user. When there are no conflicts and all regions are coloured, the game automatically triggers the Victory state.

### **Timer and scoring**

- The timer starts when the player colours the first region.
- The timer stops immediately upon a valid complete colouring.
- The time is displayed in minutes and seconds.
- The final time is stored in the Hall of Fame file (along with player name and difficulty level).

### **Hall of Fame**

The Hall of Fame is stored in a plain text file named scores.txt. Each line contains one record in the following format: <player\_name> <time\_in\_seconds>

The list is sorted in ascending order by completion time. If two records have the same time, they are sorted alphabetically by player name. If the file does not exist, it is automatically created when the first score is saved.

#### Rank | Player | Difficulty | Time

1 | Ivan | 1:23

2 | Peter | 1:41

When the Victory screen appears, the user can type their name (letters, numbers, backspace, enter). On pressing Enter, the result is saved, and the Hall of Fame screen appears.

#### Expected output

##### Normal gameplay

- Regions change colour instantly after clicks.
- Conflicts appear visually (brighter shade).
- Timer runs visibly on HUD.
- Current colour and difficulty always displayed.

##### Victory

- Message: "Congratulations! You completed the map in X seconds."
- Input prompt: "Enter your name." (keyboard input).
- On Enter → saved to Hall of Fame → Hall of FAME shown(sorted by time and difficulty)

##### Program exit

- Clean window closure.
- Scores saved persistently to file.

#### Error handling and edge cases

- |   |  |
|---|--|
| ○ <b>Clicking outside the map</b>               | No effect.                             |
| ○ <b>Clicking same region/colour repeatedly</b> | No change.                             |
| ○ <b>Deleting or missing score file</b>         | Automatically recreated on first save. |
| ○ <b>Invalid characters in name input</b>       | Ignored (only printable ASCII 32–126). |
| ○ <b>All regions same colour</b>                | Allowed, but conflicts will appear.    |
| ○ <b>No colour chosen</b>                       | Default: first colour in palette.      |

## **Visual style**

- **Map:** filled irregular regions, black borders.
- **Palette:** 4 visible colour squares with thick border.
- **Conflicts:** same colour but 30–40% brighter.
- **Background:** dark grey (to contrast regions).