Electric Field Visualizer

KIV-UPG 2024/25 Semester Project
1st Submission, Hours spent: 18

Jakub Vokoun, A23B023P

October 2024

Contents

1 Using Program

The program is only tested and anticipated for Windows.
Double-click *Build.cmd* or run it via the command-line to build it:

```
C:/Folder>Build.cmd
```

Then double-click *Run.cmd* to open the program with the default scenario. If you want to specify a scenario, pass it to the command line as follows:

```
C:/Folder>Run.cmd S
```

Where 'S' is the name of the desired scenario, or you could use name.json

2 Basic Functionality

Opening the program shows a window with visualization containing particles (circles) and probe (arrow). You can open other windows from the menu and see statistics or customize the visuals. To close the program, simply click the **x** in the upper right corner, or press *Ctrl + W*.

2.1 Particles

Particles are loaded from scenario and saved as class instances of Particle with attributes: X, Y, value. They are shown with different color for positive and negative values and are bigger for bigger value.
Particle can only have static value (in Coulomb) and position.

2.2 Probe

Probe is saved as instance of a Probe class, it travels in circle as defined in assignment (defined radius and angular velocity). It is shown as an arrow with dynamic length (linear scaling), with max and min size.

3 Implementation

The whole program follows MVC architecture, it's divided into three folders: Model, View, Controller — and also folder Scenario.

3.1 Model

Contains classes Probe, Particle and also classes FieldCalculator and Utils.
The first two encapsulate information. The second two are helper classes.

3.2 Controller

Has classes MainForm, Scenario and Program. Program is the entry point to the application, parses the command-line input, and then passes the control to the MainForm.
Scenario loads scenarios from the folder Scenario. The scenarios are saved as JSON, and you can load any scenario by its name. If any error is encountered, MessageBox with the error message is shown and the default scenario (0.json) is loaded.
MainForm initializes MVC, other windows and is responsible for time management. It also stored all important information as its attributes.

3.3 View

Has two main classes, DrawingPanel and Renderer. Alongside them are two forms: StatsForm and CustomizerForm. Drawing panel is just an encapsulation.
The two form classes shows and act upon individual windows. The Stats form just displays information, while the Customizer form has some Events because it can change a few things in the main visualization.
The Renderer class does the rendering, has functions for setting scaling, drawing Particle and drawing Probe. It also contains the function for calculating the *Drawing coordinates* from Real coordinates and is in charge of showing Customizer form and responding to its Events.

4 Tidbits

- The Drawing panel is re-drawn every Tick — set by *fps* in MainForm.
- The Probe value is shown with two-digit precision to be more user-friendly.
- The Probe length is scaled linearly but is clamped within min and max value to avoid errors, but from some value it doesn't visually display the change. Will maybe use Logarithmic scaling in the future?
- Every window can be closed via *Ctrl + W*.
- The scaling function has *padding* incorporated, so the program looks more modern.

5 Most important functions

5.1 MainForm init

This function by calling other initialize the whole application and starts the timer. Its parameter decides what scenario is used and has hard-coded the starting width and height of the window.

5.2 MainForm - OnTimerTick

This function is called every frame, calculates the difference between it was last called and now and based on this calculate the position of the Probe and invalidate the drawing. Also updates StatsForm.

5.3 Renderer - UpdateOnResize

This function is called every time the window is resized to calculate the scale, which is then stored in _scale attribute of the class Renderer. It gets the min/-max of x/y by going through all Particles and the max/min values of the Probe, e. g, (0,Radius). Then it calculates optimal scale in X and Y axis, then takes the smaller of them. Also, it sets Origin to the center of the window.

5.4 Renderer = TranslateCoordinates

Translate any given real-life coordinates to drawing coordinates, by multiplying them by scale and adding to origin coordinates. Nothing special, but one place for all translating.

5.5 Renderer - DrawProbe

Apart from drawing the Probe, it gets to dynamically calculate the length. The direction of the Probe arrow is divided by *suitable number*, than the upper and lower bounds on arrow length are set, so the arrow is visible and not out of window and by clamping and multiplying the unit vector of direction, it gets the end-point of the Probe.

6 Future

I will implement the second part of the assignment along with a few extensions which I have not chosen yet, except for the loading of scenarios.
If I decide to let the Customizer be in the program, I may implement a *Reset* button.