Web Based Visualization of 3D Medical Data using vtk.js

Sabrina Nasrin
dept. of Computing Science
University of Alberta
Edmonton, Alberta-Canada
snasrin@ualberta.ca

***Abstract*—In recent years, the development of interactive medical data solutions on the internet become only possible due to the advancement of web technologies, hence it is playing a crucial role in improving healthcare delivery to the entire society. Web-based volumetric data visualization of medical images can enhance the capability of doctors and medical researchers to diagnose diseases and to work collaboratively from any corner of the world. In this paper, we propose a web-based platform to visualize medical images using vtk.js. In this web application, user can upload their required image file and can perform volume contouring, slicing, cropping, and piecewise function by changing the colors and opacities on different levels.**

Index terms— vtk.js, MultiSliceImageMapper, ImageCropFilter, Volume Contour, Piecewise Function, Color Transfer Function.

I. Introduction

Technological advances have had a great impact on medical practice. Visualization of medical data is no longer an option—it's a must-have for modern medical organizations. Efficient 3D image visualization is crucial as the first step in the processing and analysis of medical image data. Doctors and medical researchers will be able to collaborate to perform disease diagnosis and preoperative surgery planning by web-based volumetric data visualization from anywhere around the world where the internet is available [1].

In this project, we have used the vtk.js library for visualization. vtk.js is an open-source library for interactive 2D and 3D scientific data visualization and processing. It is freely available for commercial and academic use. It can be used for volume rendering of images, slice-based rendering of images, surface renderings of geometry, point cloud rendering, VR rendering, and more. VTK.js is advantageous because of its cross-platform portability via web browsers. It can be deployed rapidly at a low cost. It is easy to visualize with vtk.js and its remote integration is flexible.

In our web application, the users can upload the image file after their choice. However, the file should be in "vti" format. "vti" is serial structured vtkImageData. vtkImageData is a data object that is a concrete implementation of vtkDataSet which represents a geometric structure that is a topological and geometrical regular array of points. Examples include volumes (voxel data) and pixmaps. This representation supports images up to three dimensions.

We have implemented volume contouring, cropping, color transfer, opacity transfer, and slicer filters in our web application in four windows on a single screen. Each functionality can be controlled separately with a control panel.

The paper is organized as follows. Section II describes the related works. Section III discusses the implementation and development. Section IV describes the author's contribution while technical challenges are given in section V. Finally, section VI draws the conclusion of this paper as well as gives an insight into the future works.

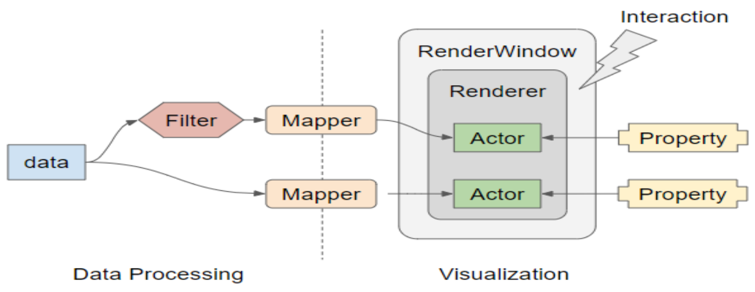

Fig. 1. Visualization Pipeline

II. Related Work

Many researchers and software companies have investigated this field in the past years. Kitware is one of them. It has been exploring web visualization for almost a decade. They have built libraries and toolkits, such as vtk.js, itk.js, paraview web, etc. for data visualization and image processing, tools for data management and analysis, and full simulation environments. They also have example applications that demonstrate how the capabilities of these tools and applications can be easily customized for specific workflows. ParaView Glance is a light-weight, open-source, VTK.js based web application developed at Kitware for visualizing volumetric images, molecular structures, geometric objects, and point clouds. ArcticViewer is a research application to explore both image-based analysis and visualization techniques for extreme-scale dataset exploration with various deployment methods ranging from standalone desktop applications to mobile application utilizing the same web core. These image-based techniques are described in greater depth in the papers [2] and [3].

III. Implementation and Development

We have used HTML as a standard markup language for documents designed to be displayed in a web browser. For our scripting language, we have used javascript in the combination of HTML and config. For module bundler, we have used webpack. Webpack takes the dependencies and generates a dependency graph allowing web developers to use a modular approach for their web application development purposes. Visualization pipeline is demonstrated in figure 1. In our project, first of all, we have implemented the filters individually with a given data file. Then we merged four filters to visualize them on a single screen in four different windows. After that, we added the upload function to the code. The upload function is for the users to choose their vti file from their local machine.

A. *Piecewise Function*

This mapping allows the addition of control points and allows the user to control the function between the control points. A piecewise hermite curve is used between control points, based on the sharpness and midpoint parameters. This function is implemented using vtkPiecewiseFunction and vtkColorTransferFunction. vtkVolumeController is used to create the controller widget for piecewise function.

B. *MultiSliceImageMapper*

Rendering process applied to 3D data where 3D images are projected to 2D surfaces by slice the image in I,J and K axis. We have kept the functionality to control the color of window and level of color with slicer. vtkImageSlice and vtkImageMapper is used to implement this filter.

C. *ImageCropFilter*

The vtkImageCropFilter will crop a vtkImageData. This will only crop against IJK-

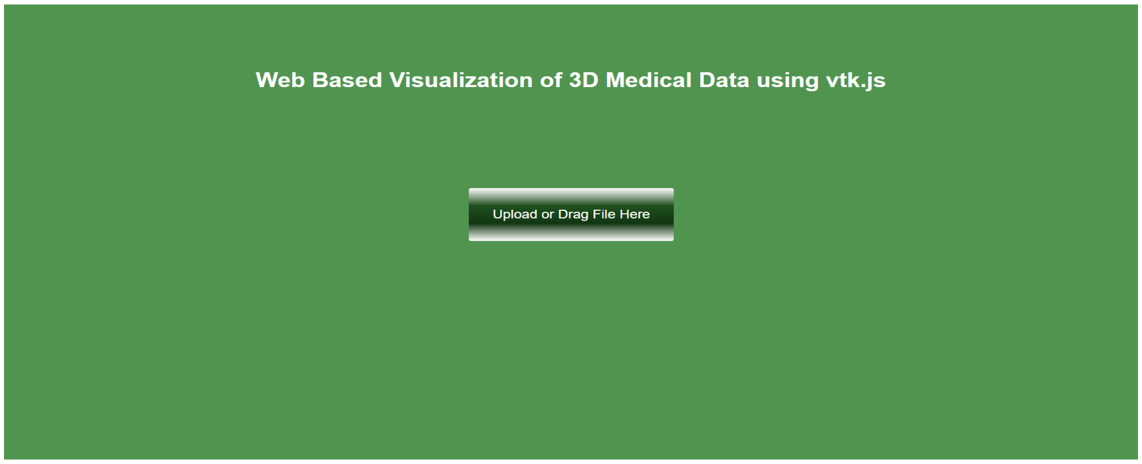

Fig. 2. Front Page.

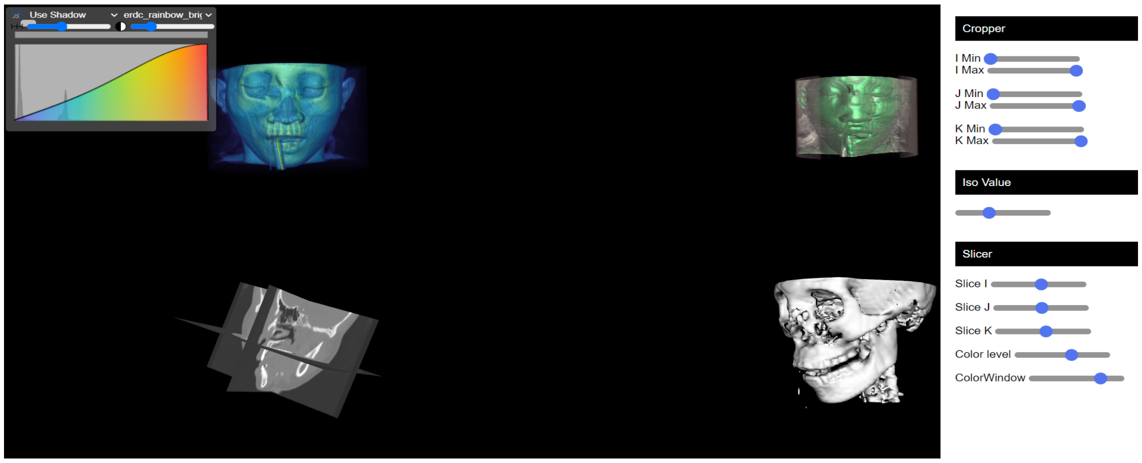

Fig. 3. Main App.

aligned planes. It is implemented using vtkPiecewiseFuntion and vtkImageCropFilter. The data file to visualize this filter is given inside the code. User can control the range of cropping in maximum and minimum level using the control panel.

D. Volume Contour

This rendering process is applied to 3D images where we vary the iso surface values for analysing inner structure of the body. This filter is implemented using vtkImageMarchingCubes. The data file to visualize this filter is also given inside the code. User can update iso value using the control panel and visualize it on the application.

The final output of our application is demonstrated in figure 2 and figure 3.

E. Tools

The tools we have used for our project are given below:

- Visual Studio Code
- @kitware/vtk.js
- node.js
- npm (in node.js install) or yarn

F. Source Code

The commented source code of our project can be found at the end of this report and here.

G. README or Compilation Instruction

- Install Visual Studio Code and Node.js
- Open terminal and write the followings:
  - mkdir my-vtkjs-app
  - cd my-vtkjs-app
  - npm init
  - npm install @kitware/vtk.js
  - npm install -D webpack-cli webpack webpack-dev-server
  - npm install –save-dev kw-web-suite
  - mkdir dist/
  - mkdir src/
- In directory dist/, create a index.html file
- In package.json file change the following lines in ”scripts”:
  - ”scripts”: { ”build”: ”webpack −−progress −−mode=development”, ”start”: ”webpack serve −−progress −−mode=development −−static=dist” }
- To run the app write ”npm run start” in terminal and navigate to ”http://localhost:8080”.

IV. Author's Contribution

My contribution in this project was to implement slicer function in the application and merge this implemented slicer with the piecewise function and upload function. User can upload their vti file from their local machine for these two functions. I have also merged other filters in the same code using function calling. I have designed the control panel and the front page of the application using html and css.

V. Technical Challenges

We faced some challenges while working on this project. As vtk.js is still developing, the resources to learn vtk.js are not rich. Only one tutorial is available on their website which is not enough to learn the working principles of vtk.js properly and also their instructions were not working in our system. We did not find any fruitful resources about vtk.js on the internet also. In addition, vtk.js is quite different from vtk python. So, we needed to spend more time solving any problems while implementing the filters for our application. However, we could solve most of the problems we faced during our project.

VI. Conclusion and Future Work

In our web application, users can upload vti files only for piecewise function and slicer. We could not merge cropper and volume contour with the upload function. In the future, we will work on merging volume contour and cropper with other filters. However, the user can visualize and control all four filters using the control panel. We were facing trouble importing the CSS file inside index.js file and merging four functionalities with the upload function. However, we could solve the problem of importing the CSS file inside index.js file. Currently, our web app is hosted on local later we can move to deploy it over the server or use Docker and Kubernetes to host our application. Now we have worked on “vti” images, in the future, we will work on reading DICOM images and data files of other formats also.

References

[1] Q. Zhang, “Web-based medical data visualization and information sharing towards application in distributed diagnosis,” *Informatics in Medicine Unlocked*, vol. 14, pp. 69–81, 2019.

[2] J. Ahrens, S. Jourdain, P. O’Leary, J. Patchett, D. H. Rogers, and M. Petersen, “An image-based approach to extreme scale in situ visualization and analysis,” in *SC ’14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 424–434.

[3] P. O’Leary, J. Ahrens, S. Jourdain, S. Wittenburg, D. H. Rogers, and M. Petersen, “Cinema image-based in situ analysis and visualization of mpas-ocean simulations,” *Parallel Computing*, vol. 55, pp. 43–48, 2016.

Source Code

Scripts	Page no
index.html	1-6
Package.json	6-7
Webpack.config.js	7
controlPanel.html	8-12
index.js [**Main Script**]	12-25
volumeViewer.css	26-28

Dist/index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css"
    />
    <style>
      /* This is for styling the control panel and the front page */
      * {
        margin: 0;
        padding: 0;
      }

      body {
        font-family: "Lato", sans-serif;
        background-color: rgb(80, 148, 80);
      }

      /* Fixed sidenav, full height */
      .sidenav {
        height: 100%;
        width: 250px;
        position: fixed;
        z-index: 1;
        top: 0;
        right: 0;
        background-color: white;
        padding-top: 20px;
        padding-left: 20px;
        margin-left: 83%;
      }
      h1 {
        text-align: center;
        margin-top: 100px;
        color: white;
      }
      .container {
        background-color: black;
        padding: 10px;
      }
      p {
        color: white;
      }
      .sliceI {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .sliceI::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .sliceJ {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .sliceJ::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
```

```css
      .sliceK {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .sliceK::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .isoValue {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .isoValue::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .colorLevel {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .colorLevel::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .colorWindow {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .colorWindow::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .Imin {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .Imin::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .Imax {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .Imax::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
```

```css
      .Jmin {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .Jmin::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .Jmax {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .Jmax::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .Kmin {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .Kmin::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
      .Kmax {
        -webkit-appearance: none;
        background: rgb(148, 147, 147);
        height: 9px;
        border-radius: 5px;
      }
      .Kmax::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 10px;
        background: rgb(82, 116, 238);
        cursor: pointer;
      }
    </style>
  </head>
  <body>
    <h1>Web Based Visualization of 3D Medical Data using vtk.js</h1>
    <script src="./main.js"></script>
  </body>
</html>
```

Package.json

```
{
  "name": "example",
  "version": "1.0.0",
  "description": "",
  "main": "src/index.js",
  "scripts": {
    "build": "webpack --progress --mode=development",
    "start": "webpack serve --progress --mode=development --static=dist",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@kitware/vtk.js": "^24.0.3",
    "curry": "^1.2.0",
    "itk": "^14.1.1"
  },
  "devDependencies": {
    "kw-web-suite": "^11.1.0",
    "webpack": "^5.71.0",
    "webpack-cli": "^4.9.2",
    "webpack-dev-server": "^4.7.4"
  }
}
```

Webpack.config.js

This is for setting the rules, input path, output path and loaders.

```
var path = require('path');
var webpack = require('webpack');
var vtkRules =
require('@kitware/vtk.js/Utilities/config/dependency.js').webpack.core.rules;


var entry = path.join(__dirname, './src/index.js');
const sourcePath = path.join(__dirname, './src');
const outputPath = path.join(__dirname, './dist');

module.exports = {
  entry,
  output: {
    path: outputPath,
    filename: 'MyWebApp.js',
  },
  module: {
    rules: [
        { test: /\.html$/, loader: 'html-loader' },
       // { test: /\.css$/, loader: 'css-loader' },
    ].concat(vtkRules),
  },
  resolve: {
    modules: [
      path.resolve(__dirname, 'node_modules'),
      sourcePath,
    ],
  },
  output: { publicPath: '/' },
};
```

Src/controlPanel.html:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css"
    />
  </head>
  <body>
    <!-- This is the html file for designing the control panel.
    Sidebar for control Panel -->
    <div class="sidenav">
      <!-- Control panel for Cropper -->
      <div class="container"><p>Cropper</p></div>
      <tr>
        <br />
        <td>I Min</td>
        <td>
          <input
            class="Imin"
            type="range"
            min="0"
            max="2.0"
            step="1"
            value="1"
          />
        </td>
        <br />
        <td>I Max</td>
        <td>
          <input
            class="Imax"
            type="range"
            min="0"
            max="2.0"
            step="1"
            value="1"
          />
        </td>
      </tr>
      <br />
      <br />
      <tr>
        <td>J Min</td>
        <td>
          <input
            class="Jmin"
            type="range"
            min="0"
            max="2.0"
            step="1"
            value="1"
          />
        </td>
        <br />
        <td>J Max</td>
        <td>
          <input
            class="Jmax"
            type="range"
            min="0"
            max="2.0"
            step="1"
            value="1"
          />
        </td>
      </tr>
      <tr>
        <br /><br />
        <td>K Min</td>
        <td>
          <input
            class="Kmin"
            type="range"
            min="0"
            max="2.0"
            step="1"
            value="1"
          />
        </td>
        <br />
        <td>K Max</td>
        <td>
          <input
            class="Kmax"
        type="range"
        min="0"
        max="2.0"
        step="1"
        value="1"
      />
    </td>
  </tr>
  <br /><br />

  <br />

  <tr>
    <td>
      <!-- Control panel for Volume Contour -->
      <div class="container"><p>Iso Value</p></div>
      <br />

      <input
        class="isoValue"
        type="range"
        min="0.0"
        max="1.0"
        step="0.05"
        value="0.0"
      />
    </td>
  </tr>
  <br /><br />

  <br />

  <tr>
    <!-- Control panel for Slicer -->
    <div class="container"><p>Slicer</p></div>
    <br />
    <td>Slice I</td>
    <td>
      <input
        class="sliceI"
        type="range"
        min="0"
        max="2.0"
        step="1"
        value="1"
      />
    </td>
  </tr>
  <br />
  <tr>
    <br />
    <td>Slice J</td>
    <td>
      <input
        class="sliceJ"
        type="range"
        min="0"
        max="2.0"
        step="1"
        value="1"
      />
    </td>
  </tr>
  <br />
  <tr>
    <br />
    <td>Slice K</td>
    <td>
      <input
        class="sliceK"
        type="range"
        min="0"
        max="100"
        step="1"
        value="1"
      />
    </td>
  </tr>
  <br />
  <tr>
    <br />
    <td>Color level</td>
    <td>
      <input
        class="colorLevel"
        type="range"
        min="-3926"
        max="3926"
        step="1"
        value="1"
        />
      </td>
    </tr>
    <br />
    <tr>
      <br />
      <td>ColorWindow</td>
      <td>
        <input
          class="colorWindow"
          type="range"
          min="0"
          max="3926"
          step="1"
          value="1"
        />
      </td>
      <br />
    </tr>
    <br />
  </div>
</body>
</html>
```

Src/index.js:

```
// Importing necessary libraries
import '@kitware/vtk.js/favicon';
import '@kitware/vtk.js/Rendering/Profiles/Volume';
import '@kitware/vtk.js/Rendering/Profiles/Geometry';
import macro from '@kitware/vtk.js/macros';
import HttpDataAccessHelper from
'@kitware/vtk.js/IO/Core/DataAccessHelper/HttpDataAccessHelper';
import vtkBoundingBox from '@kitware/vtk.js/Common/DataModel/BoundingBox';
import vtkColorTransferFunction from
'@kitware/vtk.js/Rendering/Core/ColorTransferFunction';
import vtkFullScreenRenderWindow from
'@kitware/vtk.js/Rendering/Misc/FullScreenRenderWindow';
import vtkPiecewiseFunction from
'@kitware/vtk.js/Common/DataModel/PiecewiseFunction';
import vtkVolumeController from
'@kitware/vtk.js/Interaction/UI/VolumeController';
import vtkURLExtract from '@kitware/vtk.js/Common/Core/URLExtract';
import vtkVolume from '@kitware/vtk.js/Rendering/Core/Volume';
```

```javascript
import vtkVolumeMapper from '@kitware/vtk.js/Rendering/Core/VolumeMapper';
import vtkXMLImageDataReader from '@kitware/vtk.js/IO/XML/XMLImageDataReader';
import vtkFPSMonitor from '@kitware/vtk.js/Interaction/UI/FPSMonitor';
import vtkImageCropFilter from '@kitware/vtk.js/Filters/General/ImageCropFilter';
// Force DataAccessHelper to have access to various data source
import '@kitware/vtk.js/IO/Core/DataAccessHelper/HtmlDataAccessHelper';
import '@kitware/vtk.js/IO/Core/DataAccessHelper/JSZipDataAccessHelper';
import vtkImageMapper from '@kitware/vtk.js/Rendering/Core/ImageMapper';
import vtkImageSlice from '@kitware/vtk.js/Rendering/Core/ImageSlice';
import vtkHttpDataSetReader from '@kitware/vtk.js/IO/Core/HttpDataSetReader';
import vtkImageMarchingCubes from
'@kitware/vtk.js/Filters/General/ImageMarchingCubes';
import vtkMapper from '@kitware/vtk.js/Rendering/Core/Mapper';
import vtkActor from '@kitware/vtk.js/Rendering/Core/Actor';
import  controlPanel from "./controlPanel.html";
import style from 'style-loader!css-loader?modules!./VolumeViewer.css';


let autoInit = true;
//extracting URL paramters
const userParams = vtkURLExtract.extractURLParameters();
//initializing FPS monitor
const fpsMonitor = vtkFPSMonitor.newInstance();
// ----------------------------------------------------------------------------

//This function is for removing front page after uploading the vti file
function emptyContainer(container) {
  while (container.firstChild) {
    container.removeChild(container.firstChild);
  }
}
// ----------------------------------------------------------------------------
//This preventDefault() method of the Event interface tells the user agent that
//if the event does not get explicitly handled,
//its default action should not be taken as it normally would be.
function preventDefaults(e) {
  e.preventDefault();
  e.stopPropagation();
}
// ----------------------------------------------------------------------------

//Slicer functiom

function setupMultiSlice({ renderer2, renderWindow2, source: data }) {
  //initializing slices
```

```
const imageActorI = vtkImageSlice.newInstance();
const imageActorJ = vtkImageSlice.newInstance();
const imageActorK = vtkImageSlice.newInstance();

//adding actors to the slices
renderer2.addActor(imageActorK);
renderer2.addActor(imageActorJ);
renderer2.addActor(imageActorI);

//function to update color level
function updateColorLevel(e) {
  const colorLevel = Number(
    (e ? e.target : document.querySelector(".colorLevel")).value
  );
  //setting color level property
  imageActorI.getProperty().setColorLevel(colorLevel);
  imageActorJ.getProperty().setColorLevel(colorLevel);
  imageActorK.getProperty().setColorLevel(colorLevel);
  renderWindow2.render();
}
 //function to update color window
function updateColorWindow(e) {
  const colorLevel = Number(
    (e ? e.target : document.querySelector(".colorWindow")).value
  );
  //setting color window property
  imageActorI.getProperty().setColorWindow(colorLevel);
  imageActorJ.getProperty().setColorWindow(colorLevel);
  imageActorK.getProperty().setColorWindow(colorLevel);
  renderWindow2.render();
}

//--------------------------------
//initializing mapper and setting input data, slice values and mappers
const imageMapperK = vtkImageMapper.newInstance();
imageMapperK.setInputData(data);
imageMapperK.setKSlice(30);
imageActorK.setMapper(imageMapperK);

const imageMapperJ = vtkImageMapper.newInstance();
imageMapperJ.setInputData(data);
imageMapperJ.setJSlice(30);
imageActorJ.setMapper(imageMapperJ);

const imageMapperI = vtkImageMapper.newInstance();
```

```
imageMapperI.setInputData(data);
imageMapperI.setISlice(30);
imageActorI.setMapper(imageMapperI);

//if the range of slicees are changed in the control panel input
//then the value of slices will be updated in the window also
document.querySelector(".sliceI").addEventListener("input", (e) => {
  imageActorI.getMapper().setISlice(Number(e.target.value));
  renderWindow2.render();
});

document.querySelector(".sliceJ").addEventListener("input", (e) => {
  imageActorJ.getMapper().setJSlice(Number(e.target.value));
  renderWindow2.render();
});

document.querySelector(".sliceK").addEventListener("input", (e) => {
  imageActorK.getMapper().setKSlice(Number(e.target.value));
  renderWindow2.render();
});

//get the data range
const dataRange = data.getPointData().getScalars().getRange();
const extent = data.getExtent();

//setting attribute for values from control panel input
[".sliceI", ".sliceJ", ".sliceK"].forEach((selector, idx) => {
  const el = document.querySelector(selector);
  el.setAttribute("min", extent[idx * 2 + 0]);
  el.setAttribute("max", extent[idx * 2 + 1]);
  el.setAttribute("value", 30);
});
 //setting attribute for values from control panel input
[".colorLevel", ".colorWindow"].forEach((selector) => {
  document.querySelector(selector).setAttribute("max", dataRange[1]);
  document.querySelector(selector).setAttribute("value", dataRange[1]);
});
document
  .querySelector(".colorLevel")
  .setAttribute("value", (dataRange[0] + dataRange[1]) / 2);
updateColorLevel();
updateColorWindow();
//if color level is changed then call updateColorLevel function
document
  .querySelector(".colorLevel")
```

```
    .addEventListener("input", updateColorLevel);
  document
    .querySelector(".colorWindow")
    .addEventListener("input", updateColorWindow);

  //making the slices global
  global.imageActorI = imageActorI;
  global.imageActorJ = imageActorJ;
  global.imageActorK = imageActorK;


  renderer2.resetCamera();
  renderWindow2.render();
}

//end of setupmultislice


// cropper function
function setupCropper({renderer3,renderWindow3}){
  //setting up control panel
  function setupControlPanel(data, cropFilter) {
    const axes = ['I', 'J', 'K'];
    const minmax = ['min', 'max'];

    const extent = data.getExtent();
    //setting the attributes with the change of inputs from the control panel
    axes.forEach((ax, axi) => {
      minmax.forEach((m, mi) => {
        const el = document.querySelector(`.${ax}${m}`);
        el.setAttribute('min', extent[axi * 2]);
        el.setAttribute('max', extent[axi * 2 + 1]);
        el.setAttribute('value', extent[axi * 2 + mi]);

        el.addEventListener('input', () => {
          const planes = cropFilter.getCroppingPlanes().slice();
          planes[axi * 2 + mi] = Number(el.value);
          cropFilter.setCroppingPlanes(...planes);
          renderWindow3.render();
        });
      });
    });
  }

  // creating filter
  const cropFilter = vtkImageCropFilter.newInstance();
  const reader = vtkHttpDataSetReader.newInstance({ fetchGzip: true });
  //creating pipeline
  const actor = vtkVolume.newInstance();
  const mapper = vtkVolumeMapper.newInstance();
  mapper.setSampleDistance(1.1);
  actor.setMapper(mapper);

  // create color and opacity transfer functions
  const ctfun = vtkColorTransferFunction.newInstance();
  ctfun.addRGBPoint(0, 85 / 255.0, 0, 0);
  ctfun.addRGBPoint(95, 1.0, 1.0, 1.0);
  ctfun.addRGBPoint(225, 0.66, 0.66, 0.5);
  ctfun.addRGBPoint(255, 0.3, 1.0, 0.5);
  const ofun = vtkPiecewiseFunction.newInstance();
  ofun.addPoint(0.0, 0.0);
  ofun.addPoint(255.0, 1.0);
  actor.getProperty().setRGBTransferFunction(0, ctfun);
  actor.getProperty().setScalarOpacity(0, ofun);
  actor.getProperty().setScalarOpacityUnitDistance(0, 3.0);
  actor.getProperty().setInterpolationTypeToLinear();
  actor.getProperty().setUseGradientOpacity(0, true);
  actor.getProperty().setGradientOpacityMinimumValue(0, 2);
  actor.getProperty().setGradientOpacityMinimumOpacity(0, 0.0);
  actor.getProperty().setGradientOpacityMaximumValue(0, 20);
  actor.getProperty().setGradientOpacityMaximumOpacity(0, 1.0);
  actor.getProperty().setShade(true);
  actor.getProperty().setAmbient(0.2);
  actor.getProperty().setDiffuse(0.7);
  actor.getProperty().setSpecular(0.3);
  actor.getProperty().setSpecularPower(8.0);

  //reading the data
  reader.setUrl(`https://kitware.github.io/vtk-
js/data/volume/headsq.vti`).then(() => {
    reader.loadData().then(() => {
      renderer3.addVolume(actor);

      const data = reader.getOutputData();

      //set up crop planes
      cropFilter.setCroppingPlanes(...data.getExtent());

      //calling setupControlPanel function
      setupControlPanel(data, cropFilter);
      //set up interactor, camera and renderwindow
      const interactor = renderWindow3.getInteractor();
      interactor.setDesiredUpdateRate(15.0);
      renderer3.resetCamera();
      renderWindow3.render();

    });
  });

  //set up input connection
  cropFilter.setInputConnection(reader.getOutputPort());
  mapper.setInputConnection(cropFilter.getOutputPort());

  //making the variables global
  global.source = reader;
  global.mapper = mapper;
  global.actor = actor;
  global.ctfun = ctfun;
  global.ofun = ofun;
  global.renderer = renderer3;
  global.renderWindow = renderWindow3;
  global.cropFilter = cropFilter;
}
//end of cropper

//Volume Contour function

function setupIso({renderer4,renderWindow4}){

  //creating pipeline
  const actor2 = vtkActor.newInstance();
  const mapper2 = vtkMapper.newInstance();
  //initializing marchingcube
  const marchingCube = vtkImageMarchingCubes.newInstance({
    contourValue: 0.0,
    computeNormals: true,
    mergePoints: true,
  });

actor2.setMapper(mapper2);
mapper2.setInputConnection(marchingCube.getOutputPort());
 //function to update isoValue
function updateIsoValue(e) {
  const isoValue = Number(e.target.value);
  marchingCube.setContourValue(isoValue);
  renderWindow4.render();
}
  //reading the data
  const reader = vtkHttpDataSetReader.newInstance({ fetchGzip: true });
  marchingCube.setInputConnection(reader.getOutputPort());
  const __BASE_PATH__ = 'https://kitware.github.io/vtk-js';
  reader
    .setUrl(`${__BASE_PATH__}/data/volume/headsq.vti`, { loadData: true })
    .then(() => {
      const data = reader.getOutputData();
      //setting the datarange and and isoValue
      const dataRange = data.getPointData().getScalars().getRange();
      const firstIsoValue = (dataRange[0] + dataRange[1]) / 3;

      //setting attributes for the input change in the control panel
      const el = document.querySelector('.isoValue');
      el.setAttribute('min', dataRange[0]);
      el.setAttribute('max', dataRange[1]);
      el.setAttribute('value', firstIsoValue);
      el.addEventListener('input', updateIsoValue);
      marchingCube.setContourValue(firstIsoValue);
     //setting the camera and renderwindow
      renderer4.addActor(actor2);
      renderer4.getActiveCamera().set({ position: [1, 1, 0], viewUp: [0, 0, -1]
});
      renderer4.resetCamera();
      renderWindow4.render();
    });
  //making some variables global
  global.actor = actor2;
  global.mapper = mapper2;
  global.marchingCube = marchingCube;
}

//end of volume contour
//--------------------------------------------------------------------------------

//creating screen for the application
function createViewer(rootContainer, fileContents, options) {
  //creating 4 divs' for four different windows
  const div1 = document.createElement("div");
  document.body.appendChild(div1);
  const div2 = document.createElement("div");
  document.body.appendChild(div2);
```

```
 const div3 = document.createElement("div");
 document.body.appendChild(div3);
 const div4 = document.createElement("div");
 document.body.appendChild(div4);

 //setting background color
 const background = options.background
   ? options.background.split(",").map((s) => Number(s))
   : [0, 0, 0];
 const containerStyle = options.containerStyle;

 //initializing screen renderer for 4 windows
 const fullScreenRenderer = vtkFullScreenRenderWindow.newInstance({
   background,
   rootContainer:div1,
   containerStyle
 });
 const fullScreenRenderWindow = vtkFullScreenRenderWindow.newInstance({
   background: [0, 0, 0],
 },
 { rootContainer: div2    }
 );
 //slicer
 const fullScreenRender1 = vtkFullScreenRenderWindow.newInstance({
   background: [0, 0, 0],
 },
 { rootContainer: div3}
 );
 const fullScreenRender2 = vtkFullScreenRenderWindow.newInstance({
   background: [0, 0, 0],
 },
 { rootContainer: div4}
 );
 const renderer = fullScreenRenderer.getRenderer();
 const renderWindow = fullScreenRenderer.getRenderWindow();
 renderWindow.getInteractor().setDesiredUpdateRate(15);

//setting position and sizes of the windows
 //piecewise function window
 fullScreenRenderer.getContainer().style.height = "50%"
 fullScreenRenderer.getContainer().style.width = "50%"
 fullScreenRenderer.getContainer().style.right = "50%"
 fullScreenRenderer.getContainer().style.bottom= "50%"
 fullScreenRenderer.resize()
 //cropper window
```

```
  fullScreenRenderWindow.getContainer().style.height = "50%"
  fullScreenRenderWindow.getContainer().style.width = "50%"
  fullScreenRenderWindow.getContainer().style.left= "50%"
  fullScreenRenderWindow.getContainer().style.bottom= "50%"
  fullScreenRenderWindow.resize()
  const renderer3 = fullScreenRenderWindow.getRenderer();
  const renderWindow3 = fullScreenRenderWindow.getRenderWindow();

  //slicer
  fullScreenRender1.getContainer().style.height = "50%"
  fullScreenRender1.getContainer().style.width = "50%"
  fullScreenRender1.getContainer().style.right = "50%"
  fullScreenRender1.getContainer().style.top ="50%"
  fullScreenRender1.resize()
  const renderer2 = fullScreenRender1.getRenderer();
  const renderWindow2 = fullScreenRender1.getRenderWindow();

 //volume contour
  fullScreenRender2.getContainer().style.height = "50%"
  fullScreenRender2.getContainer().style.width = "50%"
  fullScreenRender2.getContainer().style.left = "50%"
  fullScreenRender2.getContainer().style.top ="50%"
  fullScreenRender2.resize()
  const renderer4 = fullScreenRender2.getRenderer();
  const renderWindow4 = fullScreenRender2.getRenderWindow();

  //reading the image from local machine
  const vtiReader =vtkXMLImageDataReader.newInstance({ fetchGzip: true });
  vtiReader.parseAsArrayBuffer(fileContents);

   //creating the pipeline
  const source = vtiReader.getOutputData();
  const mapper = vtkVolumeMapper.newInstance();
  const actor = vtkVolume.newInstance();

  const dataArray =
    source.getPointData().getScalars() || source.getPointData().getArrays()[0];
  const dataRange = dataArray.getRange();

  //initializing the color transfer and opacity transfer function
  //and adding the control panel
  const lookupTable = vtkColorTransferFunction.newInstance();
  const piecewiseFunction = vtkPiecewiseFunction.newInstance();
  fullScreenRenderer.addController(controlPanel);
```

```
// Pipeline handling
actor.setMapper(mapper);
mapper.setInputData(source);
renderer.addActor(actor);

//calling the other functions: slicer, cropper and volume contour
setupMultiSlice({ source, renderer2, renderWindow2 });
setupCropper({renderer3,renderWindow3});
setupIso({renderer4,renderWindow4});

// Configuration
const sampleDistance =
  0.7 *
  Math.sqrt(
    source
      .getSpacing()
      .map((v) => v * v)
      .reduce((a, b) => a + b, 0)
  );
mapper.setSampleDistance(sampleDistance);
actor.getProperty().setRGBTransferFunction(0, lookupTable);
actor.getProperty().setScalarOpacity(0, piecewiseFunction);
actor.getProperty().setInterpolationTypeToFastLinear();
actor.getProperty().setInterpolationTypeToLinear();

// For better looking volume rendering
// - distance in world coordinates a scalar opacity of 1.0
actor
  .getProperty()
  .setScalarOpacityUnitDistance(
    0,
    vtkBoundingBox.getDiagonalLength(source.getBounds()) /
      Math.max(...source.getDimensions())
  );


actor.getProperty().setGradientOpacityMinimumValue(0, 0);
actor
  .getProperty()
  .setGradientOpacityMaximumValue(0, (dataRange[1] - dataRange[0]) * 0.05);

//setting shading based on gradient
actor.getProperty().setShade(true);
actor.getProperty().setUseGradientOpacity(0, true);
```

```
//setting the deafault values
actor.getProperty().setGradientOpacityMinimumOpacity(0, 0.0);
actor.getProperty().setGradientOpacityMaximumOpacity(0, 1.0);
actor.getProperty().setAmbient(0.2);
actor.getProperty().setDiffuse(0.7);
actor.getProperty().setSpecular(0.3);
actor.getProperty().setSpecularPower(8.0);

//initializing control widget and UI
const controllerWidget = vtkVolumeController.newInstance({
  size: [280, 150],
  rescaleColorMap: true
});
const isBackgroundDark = background[0] + background[1] + background[2] < 1.5;
controllerWidget.setContainer(rootContainer);
 // setUpContent sets the size to the container.
controllerWidget.setupContent(renderWindow, actor, isBackgroundDark);

fullScreenRenderer.setResizeCallback(({ width, height }) => {
  controllerWidget.render();
  fpsMonitor.update();
});

// First render
renderer.resetCamera();
renderWindow.render();

global.pipeline = {
  actor,
  renderer,
  renderWindow,
  lookupTable,
  mapper,
  source,
  piecewiseFunction,
  fullScreenRenderer
};

if (userParams.fps) {
  const fpsElm = fpsMonitor.getFpsMonitorContainer();
  fpsElm.classList.add(style.fpsMonitor);
  fpsMonitor.setRenderWindow(renderWindow);
  fpsMonitor.setContainer(rootContainer);
  fpsMonitor.update();
}
```

```
}

// ---------------------------------------------------------------------------
// loading data function and checking whether the file is vti or not
export function load(container, options) {
  autoInit = false;
  emptyContainer(container);

  if (options.file) {
    if (options.ext === "vti") {
      const reader = new FileReader();
      reader.onload = function onLoad(e) {
        createViewer(container, reader.result, options);
      };
      reader.readAsArrayBuffer(options.file);
    } else {
      console.error("Unkown file...");
    }
  }

}

//this function is controlling the front page function.
//It is adding the vti file after clicking the upload button
//after uploading transfering the data to the second page that is the main
application page
export function initLocalFileLoader(container) {
  const exampleContainer = document.querySelector(".content");
  const rootBody = document.querySelector("body");
  const myContainer = container || exampleContainer || rootBody;

  const fileContainer = document.createElement("div");
  fileContainer.innerHTML = `<button class = "${style.dropButton}">Upload or Drag
File Here</button><div class="${style.bigFileDrop}"/><input type="file"
accept=".vti" style="display: none;"/>`;
  myContainer.appendChild(fileContainer);

  const fileInput = fileContainer.querySelector("input");
  function handleFile(e) {
    preventDefaults(e);
    const dataTransfer = e.dataTransfer;
    const files = e.target.files || dataTransfer.files;
    if (files.length === 1) {
      myContainer.removeChild(fileContainer);
      const ext = files[0].name.split(".").slice(-1)[0];
      const options = { file: files[0], ext, ...userParams };
      load(myContainer, options);
    }
  }
  //these are codes for how the data file can be added
  fileInput.addEventListener("change", handleFile);
  fileContainer.addEventListener("drop", handleFile);
  fileContainer.addEventListener("click", (e) => fileInput.click());
  fileContainer.addEventListener("dragover", preventDefaults);
}


//it is loading the container with the parameters from the loaded data file
if (userParams.fileURL) {
  const exampleContainer = document.querySelector(".content");
  const rootBody = document.querySelector("body");
  const myContainer = exampleContainer || rootBody;
  load(myContainer, userParams);
}

//loading the data file
const viewerContainers = document.querySelectorAll(".vtkjs-volume-viewer");
let nbViewers = viewerContainers.length;
while (nbViewers--) {
  const viewerContainer = viewerContainers[nbViewers];
  const fileURL = viewerContainer.dataset.url;
  const options = {
    containerStyle: { height: "100%" },
    ...userParams,
    fileURL
  };
  load(viewerContainer, options);
}

// Auto setup if no method get called within 100ms
setTimeout(() => {
  if (autoInit) {
    initLocalFileLoader();
  }
}, 100);
```

Src/volumeViewer.css

```css
.fullScreen {
    position: absolute;
    width: 100vw;
    height: 100vh;
    top: 0;
    left: 0;
    overflow: hidden;
    background: black;
    margin: 0;
    padding: 0;
  }
.fullParentSize {
    position: absolute;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    overflow: hidden;
  }
.dropButton{

    font-size: larger;
    display: flex;
    align-items: center;
    justify-content: center;
    margin-left:41%;
    margin-top: 150px;
    margin-bottom: 250px;
    background-image: linear-gradient(white,rgb(31, 80, 31), rgb(18, 54,
18),white);
    border: none;
    color: white;
    padding: 30px 32px;
    cursor: pointer;
    border-radius: 3px;
}

.bigFileDrop {

  cursor: pointer;

}
```

```css
.selector {
  position: absolute;
  top: 5px;
  left: 400px;
  transform: translate(-100%, 0px);
  border: none;

  color: green;
  border: none;
}

select {
  -moz-appearance: none;
}

select option {
  color: black;
}

select:focus {
  outline: none;
  border: none;
}

.piecewiseWidget {
  position: absolute;
  top: calc(10px + 1em);
  left: 5px;
  background: rgba(255, 255, 255, 0.3);
  border-radius: 5px;
}

.shadow {
  position: absolute;
  top: 5px;
  left: 5px;
  border: none;
  background: transparent;
  color: white;
  border: none;
}

.progress {
  flex: none;
  font-size: 50px;
  color: black;
  z-index: 1;
  background: rgba(128,128,128,.5);
  padding: 20px;
  border-radius: 10px;
  user-select: none;
}

.fpsMonitor {
  position: absolute;
  bottom: 10px;
  left: 10px;
  background-color: rgba(255, 255, 255, 0.5);
  border-radius: 5px;
  border: solid 1px gray;
}
```