**גרפיקה ממוחשבת – תרגיל 3:**

**Modules:**

1. **App** 
   * **Path:** './includes/js/app.js'
   * **Role:** responsible for managing the global state of the canvas app.
   * **Functions:**
     + **initialize() : void**
       - create references to the Canvas' html object.
     + **loadInput() : void** 🡪 initializePolygons() : void
     + **initializePolygons(input) : void**
       - sets the global state of points & polygons according to the input data.
       - Randomize array[polygons.length] of colors.
       - Normalize points()
       - For each polygon: create new drawable Polygon class and saves them in App.draws.polygons[]
     + **normalizePoints() : void**
       - convert points from: world coords 🡪 local coords.
     + **drawPolygons() : void**
       - clearPolygons()
       - doProjection() //depends on App.settings.projection state
       - for each Polygon in App.draws.polygons[] 🡪 Polygon.draw()
     + **doProjection() : void**
       - call projection function according to: App.settings.projection:
         * projections: oblique(deg) | perspective() | orthographic()
     + **clearPolygons() : void**
       - clear Canvas
       - for each Polygon in App.draws.polygons[] 🡪 Polygon.syncPoints()
2. **Transforms (transforms.js)**
   * **Path:** './includes/js/transform.js'
   * **Role:** responsible for all the transformations.
   * **Functions:**
     + **Projections:** By Polygon Class on it's own points[]
       - **orthographic() : void**
         * For each Polygon in App.draws.polygons[] 🡪 Polygon.orthographic()
       - **perspective() : void**
         * For each Polygon in App.draws.polygons[] 🡪 Polygon.perspective()
       - **oblique(deg) : void**
         * For each Polygon in App.draws.polygons[] 🡪 Polygon.oblique(deg)
     + **Global Transforms:** by App on global points[]
       - **syncPoints() : void**
         * for each drawable Polygon: Polygon.syncPoints()
         * it will sync each polygins local points like [1,4,7,12] with their current global\_points value.
       - **scale(k) : void**
         * scale each global\_point by k
         * syncPoints()
       - **rotateX() | rotateY() | rotateZ() : void**
         * for each global\_point:

convert to it's according Vector.

Multiple it by the according Matrix(X|Y|Z)

Change the global\_point value to the result of the multiply.

* + - * + syncPoints()

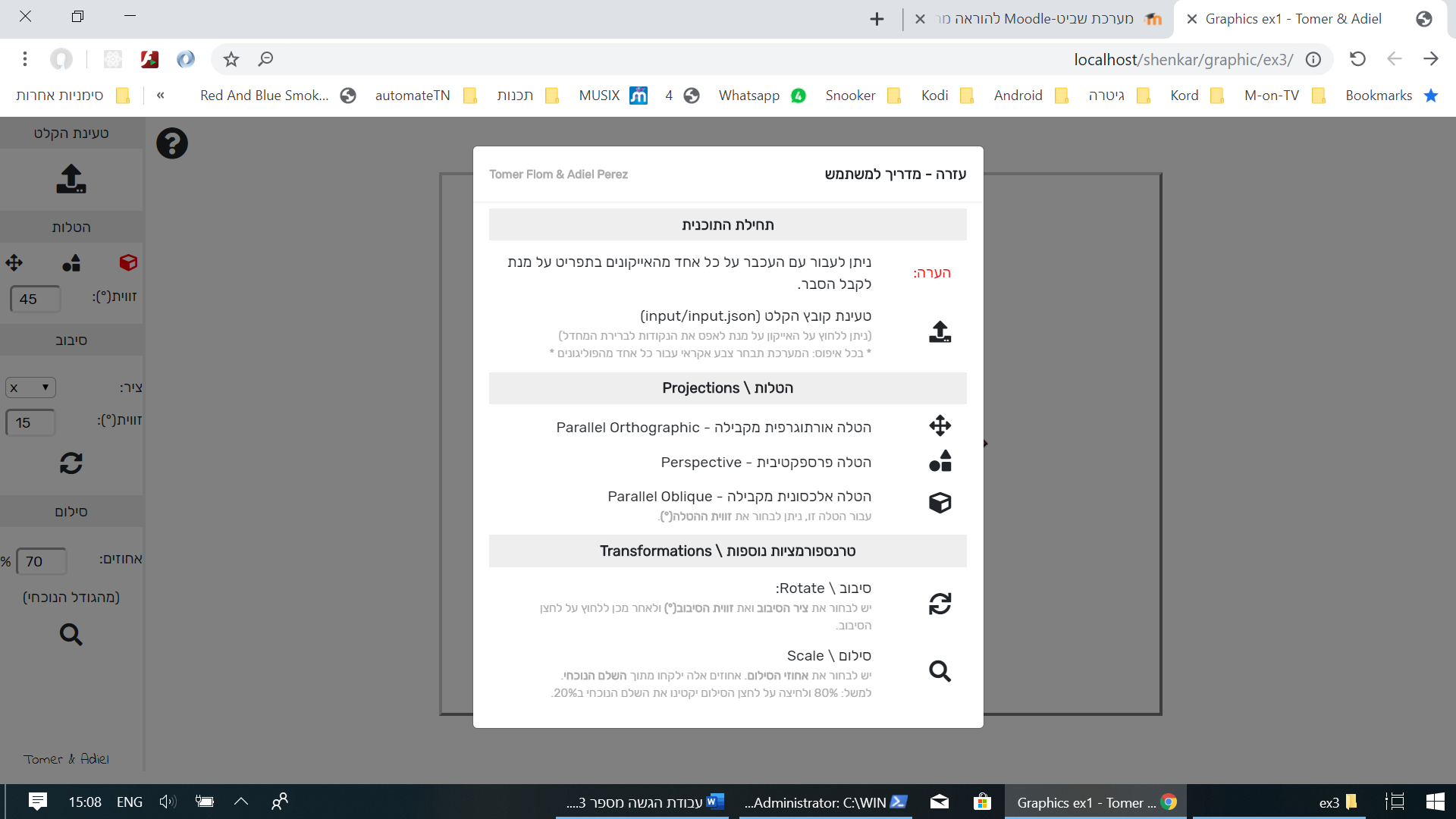
1. **Polygon : Class** 
   * **Path:** './includes/js/shapes/Polygons.js'
   * **Role:** responsible for managing the global state of the canvas app.
   * **Methods:**
     + **Constructor(points\_index[], color) : void**
       - Polygon is initialized with the **const** points\_index[] which symbolize it's own points. Example: [1,5,7,12]. This array is contstant and will never change.
       - this.syncPoints()
     + **syncPoints() : void**
       - this method will sync the local Polygon.points[] with their current value in the global\_points state.
       - this.setNormal()
     + **setNormal() : void**
       - will be called every time that Polygon.points have been changed.
       - Set this.normal according to Polygon.points.
     + **isVisible() : void**
       - return true|false according to Polygon.normal
     + **draw() : void**
       - check isVisible()
       - draw the Polygon according to Polygon.points
     + **orthographic() : void**
       - modifying the Polygon.points in orthographic algorithm
     + **perspective() : void**
       - modifying the Polygon.points in perspective algorithm
     + **oblique(deg) : void**
       - modifying the Polygon.points in oblique algorithm
2. **Misc**
   * **Path:** './includes/js/misc.js'
   * **Role:** miscellaneous functions
   * **Functions:**
     + **deg\_to\_rad(deg) : float**
     + **cos(deg) : float**
     + **sin(deg) : float**
     + **multiplyMatrix(a,b) : Matrix[][]**
     + **getRandomColorsArray(size) : String[size]**
     + **getRandomColor() : String**
3. **GUI**
   * **Path:** './includes/js/gui.js'
   * **Role:** controller for html's GUI interactions

**Errors**

1. loadInput() : getJSON('./input/input.json') fetch failed
   1. **reason:** might happens if the program isn't running on localhost environment
   2. **solution:** on case fetch failed 🡪 a static variable called INPUT (from './input/input.js') will be used.

**Screenshots:**

1. press the question-mark icon for a tutorial



1. App in use:

