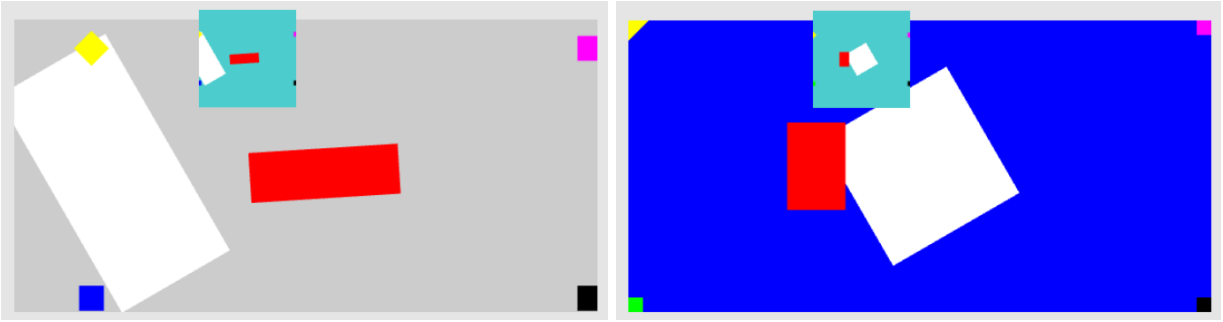
CSS 452: Programming Assignment #3
Resource Management and Scenes

Due time: Please refer to our course web-site

Objective
In this programming assignment we will work in real-time environment, expand resource_map to support JSON (another popular format) and local storage, and verify our understanding of viewport, and WC space.

Watch out for circular import!

Assignment Specification:
Here is an example of the results from this assignment:



Assignment specifications:

- **Two Scenes**: You must support at least two scenes: (please refer to this file: https://myuwbclasses.github.io/CSS452/CourseMaterials/MP3/assets.zip)
  - First Scene: Gray Scene: specified by: **scene.json**.
    - The 'N' command transits to the Blue Scene
    - The 'Q' command quits the app
  - Second Scene: Blue Scene: specified by: **blue_level.xml**
    - The 'N' command transits to the Gray Scene
- **Real time movements**: In the Gray scene, notice that:
  - **The red rectangle:** rotates at a rate of one complete revolution per 5 seconds
  - **The white rectangle:** moves towards the left and wraps around at a speed of 20 units per 3 seconds.
- **Small Viewport (in green)**: You can control the Device Coordinate (DC, or pixel positions) location of this viewport with the WASD keys.
- **Large WC Coordinate**: You can control the WC coordinate systems of the large view with the FCVB keys for translation and ZX for zooming in and out.
  - **Warning:** you will have to modify the input component to support additional key codes.
- **Input support:** Modify the input component to support "KeyReleased" event (when a key state transitions from pressed to released).
  - **Small viewport:** left-ward movement (the A-key control) is triggered by the "Key Released" event.
- **Saved game state information:** The small camera view is preserved over scene transitions. This can be confirmed by the location of the viewport for this camera: the DC location is preserved over scene transitions.

Hints:

1. My implementation is based on book Example-4.6 (*audio_support*). You do _NOT_ need to support audio in this assignment.

2. Go read up on JSON file format and how it is supported in JavaScript. The parsing is trivial. I learned how to parse JSON by examining these two sites:
   a. http://www.w3schools.com/js/js_json.asp
   b. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse

3. You should define two new engine modules: ***json*** and ***storage***. Refer to the **xml.js** module, **json.js** should be very similar, in particular, notice:
   a. ***JSON*** is very similar to XML (refer to xml.js):
      i. Decode XML file (JSON is identical, no special need to decode):
      ```
      function decodeXML(data) { return data.text(); }
      ```
      ii. Parse XML:
      ```
      let mParser = new DOMParser();
      function parseXML(text) {
          return mParser.parseFromString(text, "text/xml");
      }
      ```
      JSON can be parsed with: `JSON.parse(text);`
   b. ***storage*** define an interface to the **resource_map** module for storing and retrieving any run-time data that can persist over scene deletion/creation. This module will be similar to text/xml module, except, you don't need to load resources. Rather, you can simply store any data in your game. In this case, you will be storing the small-camera.

4. DC manipulation of viewport is simply changing the viewport location.

5. WC manipulation of camera is changing the camera location and its width.

6. Don't forget, you will have to modify ***engine/input.js*** to support the "KeyRelease" event, and, to support the additional keycodes that are required.

Credit Distribution

Here is how the credits are distributed in this assignment:

1.	Parsing JSON scene file and scene transitions		30%
	a. Define **json** module in the engine b. Parse and work with the JSON scene file c. Parse and work with the XML scene file d. Support scene transitions with "N" key	10% 20% 10% 20%	
2.	Small camera view: Viewport control		15%
	a. WASD manipulate the Viewport b. The "A" key is triggered by KeyRelease event	5% 10%	
3.	Large camera view: WC control		15%
	a. FCVB manipulate the WC Window b. ZX zooms in/out	10% 10%	
4.	Keyboard control + Speed		20%
	a. Support KeyRelease event ("A"-Key) b. Support all above keys properly c. Rotation speed (1 revolution / 5 sec) d. Movement speed (20 units / 3 sec)	10% 5% 5% 5%	
5.	Storage module (using resource_map)		15%
	a. Define **storage** module in the engine b. Small view camera Viewport is preserved over different scenes	10% 10%	
6.	Proper submission		5%
	a. Zip file names with **NO SPACES** b. No extra unused files/folders (E.g., Test folder) c. Styles (project name, variable names, etc.)	5% 5% 5%	

This programming assignment will count 11% towards your final grade for this class.

Creativity and Extra Credits: Your first two scenes MUST BE defined by the two provided files!! Sorry, but to facilitate easy grading, the first two scenes _MUST_ be the same as mine. You are free to create additional scenes if you like. BUT, the first two scenes must be identical to mine.

- Please do feel free to include your own scenes, HOWEVER, please make sure you support transitions between scenes with the N key. Also, please make sure the small view camera is preserved between Gray and Blue scenes.