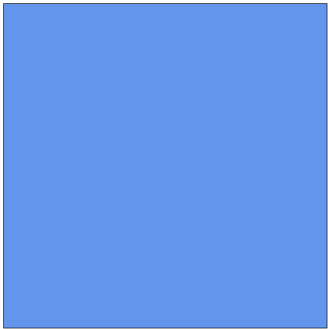
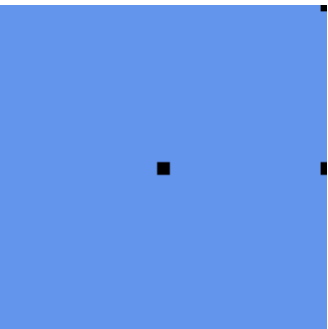
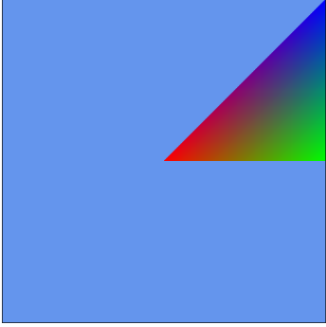

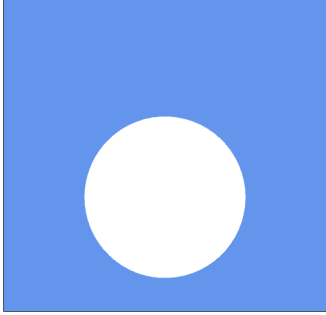


Worksheet 1: Getting started with WebGPU

Reading	RTR: Chapters 1-2. Angel: Section 3.1.
Purpose	<p>The purpose of this set of exercises is to get started with web graphics. You will set up a WebGPU application from scratch, create a canvas and a WebGPU context, compile and use simplistic shader programs, setup the needed buffers for drawing, and draw and animate simple shapes.</p> <p>A step-by-step introduction to WebGPU is available here: https://webgpufundamentals.org/</p> <p>Basic JavaScript reference: https://www.w3schools.com/jsref/</p> <p>Tips:</p> <ul style="list-style-type: none"> - You can press F12 or right click and choose “Inspect [element]” in most browsers to show the developer/debug menus which can be quite helpful.
Part 1 	Setup a basic WebGL application. <ul style="list-style-type: none"> • Create a HTML document with a 512x512 canvas element and write a script to create a WebGPU context for the canvas. • Set up a render pass and clear the canvas with the color cornflower blue (0.3921, 0.5843, 0.9294, 1.0). • If not already done, move the script to a separate JavaScript file and include it in the HTML document. In the JavaScript file, use the window.onload event to call an asynchronous main function that initializes and runs the application in the canvas.
Part 2 	Shaders and buffers. <ul style="list-style-type: none"> • Load a shader module and create a render pipeline with a vertex shader that takes 2D positions as input and a fragment shader that outputs a constant black color. • Set up a vertex buffer with 2D positions for three points (each forming a square made of two triangles). Each point should have a size of 20 pixels (width and height). Connect the vertex buffer to the position input of the vertex shader and place the points in the canvas as in the image to the left.

Worksheet 1: Getting started with WebGPU

<p>Part 3</p> 	<p>Triangles.</p> <ul style="list-style-type: none">• Change the code in the previous example to draw a triangle instead of points.• Extend the application to include a second buffer for vertex colors and draw the triangle with interpolation of red, green, and blue vertex colors.
<p>Part 4</p> 	<p>A rotating square.</p> <ul style="list-style-type: none">• Add a second triangle to the previous part such that you have a quadrilateral (which is maybe even a square).• Center your quad (short form of quadrilateral) make it of width one and height one in the default coordinate system.• Add a rotation so the quad rotates around its center. Animate the rotation angle over time. Use <code>requestAnimationFrame</code> to continuously call your render function.
<p>Part 5</p> 	<p>A bouncing circle.</p> <ul style="list-style-type: none">• Create and draw a circle using the triangle-list primitive topology.• Make the circle bounce up and down over time.