**Programmable Laser Projector**

This product will provide its target audience of 7-18 year-old students with a programmable interface that controls an RGB laser projector. For this purpose, we are working with an OEM vendor for laser projectors to develop custom hardware for us, based on their pre-existing designs plus our custom PC board design. The laser projector incorporates a pair of galvos to display real-time vector-based graphics using three lasers of different colors (red, green, and blue), thus providing full control over the color, intensity, and x-y location of the output beam with a fast refresh rate - 15kpps (kilo points per second) scanning speed.

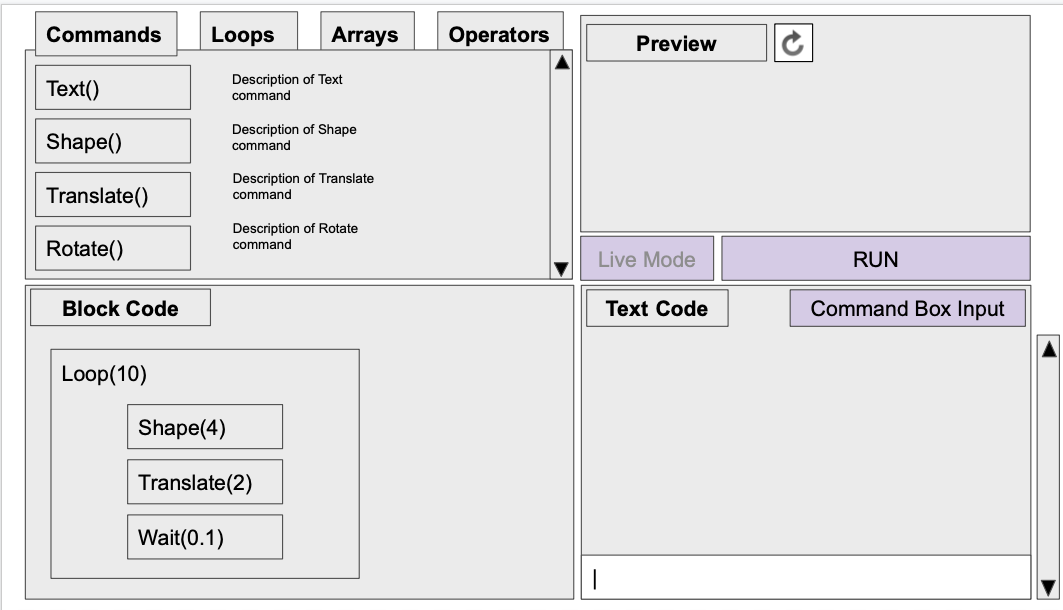
The web developer’s job is to develop a user interface that will help 7-18 year-old students develop \*translatable programming skills\* via a user-friendly programming interface. Our audience can be exemplified in two target student profiles: Profile#1 is a 7-year old student learning programming for the first time; and Profile#2 is a college-bound high-school student developing a complex artistic display. The user interface should incorporate the wow factor into programming (largely through its connection to a highly functional RGB laser-based display), while giving both categories of users translatable programming skills.

Several other points are noteworthy:

1. The web developer will not be able to test the user interface/platform with the hardware for at least a couple of weeks; it is possible the hardware will not be available to test at all. However, they need to ensure that they are meeting the required specifications.
2. The interface/platform should display well on different types of browsers: smartphone, PC, Mac, etc.
3. The platform should incorporate two parallel dialogs for inputting code:
   1. Dialog A: Novice interface (targeting Profile#1); Simple and user-friendly interface; Drag-and-drop menu using boxes for the commands, etc.
   2. Dialog B: Advanced interface (targeting Profile#2); More options for them; Higher difficulty but more capabilities; Command-line / text-based interface.

Dialog A should be based on a drag-and-drop interface using [Blockly](https://developers.google.com/blockly) (or something similar), with the output making calls to a WebSocket. Dialog B should be based on a text-based interface using [Skulpt](https://skulpt.org/) and custom Skulpt commands (or an equivalent). Code created via the drag-and-drop interface in Dialog A should also automatically display the corresponding Python code (via a built-in Python function) in Dialog B so the student can start to become familiar with it. The interface should provide options for “Live” mode (where the laser projector immediately executes the code as each line is entered) and “Run” mode (where the laser projector only executes the code when the user clicks on the RUN button). If possible, the user interface should also include an output preview where the laser output is represented on the screen of their device. Also, the frontend should communicate with the hardware via websocket, though there is an interest in communicating via webRTC or UDP hole punching in the long term.

To summarize, the student should be able to create a Python-based program using Blockly (in Dialog A) or Skulpt (in Dialog B); choose between “Live” or “Run” modes; review the Python code in Dialog B (regardless of which Dialog was used to input the code); view an output preview on the screen of their device (if possible); and execute the code to display the result on the laser projector by clicking “RUN” (if in “Run” mode).

Example of interface: 

The user interface needs to enable the student to complete all of the exercises defined in the following 10 Modules (detailed descriptions to follow):

**Module 1: INTRO  
Module 2: LOOPS  
Module 3: CAPTURE INPUT  
Module 4: CONDITIONAL STATEMENTS  
Module 5: ARRAYS  
Module 6: ARRAY OPERATORS  
Module 7: RECURSION   
Module 8: DICTIONARIES   
Module 9: OBJECT ORIENTED PROGRAMMING  
Module 10: STATE MACHINES**

**MODULE 1: INTRO**

**Exercise 1.1: “Hello World”**

* Activity: Output text, Select Color + Size + Display style in time domain
  1. Input text ⇒ compile ⇒ observe results
  2. Change color ⇒ compile ⇒ observe results
  3. Change size ⇒ compile ⇒ observe results
  4. Change display style ⇒ compile ⇒ observe results
* Learning goals:
  1. Understand programming interface
  2. Demonstrate ability to control the laser projector using pre-programmed functions to display text

**Exercise 1.2: “Shapes”**

* Activity: Output basic geometric shapes, Select Color + Size + Animation style
  1. Input shape type using name (point, line, triangle, rectangle, pentagon, hexagon, septagon, octagon, nonagon, decagon, circle) ⇒ compile ⇒ observe results
  2. Input shape type using # vertices (1,2,3,4,5,6,7,8,9,10,inf) ⇒ compile ⇒ observe results
  3. Change color ⇒ compile ⇒ observe results
  4. Change size ⇒ compile ⇒ observe results
  5. Change display style ⇒ compile ⇒ observe results
* Learning goals:
  1. Understand how to use different commands to display different shapes
  2. Learn that different functions - text and shape - use some common parameters

**MODULE 2: LOOPS**

**Exercise 2.1: “Movement”**

* Activity: Output shape or text, apply transform to rotate or translate, repeat command using loop to create movement
  1. Input infinite loop containing {shape type + motion type (translate or rotate)} ⇒ compile ⇒ observe results
  2. Select finite loop ⇒ compile ⇒ observe results
  3. Change motion type ⇒ compile ⇒ observe results
  4. Change motion direction ⇒ compile ⇒ observe results
  5. Change motion speed ⇒ compile ⇒ observe results
  6. Change shape to text ⇒ compile ⇒ observe results
* Learning goals:
  1. Learn how to use loops with commands to create different kinds of motion
  2. Understand that motion controls can be used with both shapes and text

**Exercise 2.2: “Random Walk”**

* Activity: Output cartoon figure with walking animation, and move figure in random walk
  1. Select cartoon figure and speed of walking animation ⇒ compile ⇒ observe results
  2. Change speed of walking animation ⇒ compile ⇒ observe results
  3. Change cartoon figure ⇒ compile ⇒ observe results
  4. Create an infinite loop that moves the figure by one step and waits, then repeats
  5. Add a second motion step in the loop, so the figure takes a step to the right and waits and takes a step towards the ceiling and waits again, then repeats
  6. Add 4 more steps to the loop, so the figure loops through 6 steps and appears to travel a random walk
* Learning goals:
  1. Learn that multiple actions can be placed within one loop
  2. Understand the limitations of simulated random patterns

**Exercise 2.3: “Chirping birds”**

* Activity: Output birds that flap wings, move, and chirp using nested loops.
  1. Select bird type ⇒ compile ⇒ observe results
  2. Change bird type ⇒ compile ⇒ observe results
  3. Change chirp type ⇒ compile ⇒ observe results
  4. Change wing motion speed ⇒ compile ⇒ observe results
  5. Input infinite loop containing {display bird + translation motion type + nested finite loop containing {flap wings + chirp}}
  6. Change bird motion speed ⇒ compile ⇒ observe results
  7. Change number of chirps at a time ⇒ compile ⇒ observe results
* Learning goals:
  1. Learn how to use nested loops
  2. Learn how to create multimedia animations using pre-existing functions

**MODULE 3: CAPTURE INPUT**

**Exercise 3.1: “Random Walk v2”**

* Activity: Modify code from Exercise 2.2 so that the code captures user input as a queue to continue the random walk
  1. Replace the last waiting function in the loop with blocking code that queues the user to press a key to initiate the next step
  2. Modify the code so that the 6 lines of code used to move the figure are replaced by a nested loop that iterates 3 times; inside the nested loop, the figure should take one horizontal step, wait, take one vertical step, and wait; use the random function to set the step size for each step, with a range from [-10,10]
* Learning goals:
  1. Learn how to use blocking code to wait for user input
  2. Learn how to use the random function

**Exercise 3.2: “Turtle”**

* Activity: Output drawing turtle, and draw one step of a staircase with each user input, using the input value as the step size
  1. Inside an infinite loop, output a triangle that rotates to the right, then moves one step to the right, then waits, then rotates to face up, then moves one step up, then waits; observe as the “turtle” repeatedly traces out a staircase
  2. Add user input as blocking code, so that the user is required to input a numeric value after each step, and that value is used as the size of the next two steps
  3. Add a line drawing command to the loop so that the path traveled by the turtle is traced out in a line
* Learning goals:
  1. Learn how to use variables, including defining variables, assigning values to them, calling the variables to access stored values, and performing multiplication with variables
  2. Understand the limitations of code without conditional statements

**Exercise 3.3: “Bursting Balloons”**

* Activity: Output balloons that float up through the viewing area, and that pop when the user gives a numeric input; larger number gives bigger pop
  1. Output balloons floating up through the screen, one at a time
  2. Add non-blocking user input, so that the balloon pops when the user gives a numeric input, with a bigger pop for a bigger number
  3. Change balloon size ⇒ compile ⇒ observe results
  4. Change balloon color ⇒ compile ⇒ observe results
* Learning goals:
  1. Understand how to capture user input without blocking the code
  2. Learn how to create an animation that incorporates user intervention

**MODULE 4: CONDITIONAL STATEMENTS**

**Exercise 4.1: “Chirping birds v2”**

* Activity: Modify code from Exercise 2.3 so that the user can independently turn on/off chirping, flapping of wings, and motion of bird.
  1. Create boolean variables chirpOn, flapOn, and motionOn; set them to TRUE; use non-blocking code to toggle the values of these when the user presses ‘c’, ‘f’, and ‘m’, respectively; run code and demonstrate these controls
  2. Change bird type ⇒ compile ⇒ observe results
* Learning goals:
  1. Learn how to use conditional statements
  2. Learn how to create and use boolean variables

**Exercise 4.2: “Turtle v2”**

* Activity: Modify code from Exercise 3.2 so that the user can rotate the turtle, move the turtle, and extend/retract a “pen” that will trace the path of the turtle as it moves.
  1. Output triangle that repeatedly draws a staircase without user input
  2. Add user input, so that step size is determined by input value, and user input is required after each step ⇒ compile ⇒ observe results
  3. Change shape type for turtle ⇒ compile ⇒ observe results
  4. Change color ⇒ compile ⇒ observe results
  5. Change size ⇒ compile ⇒ observe results
* Learning goals:
  1. Understand how to create interactive programs
  2. Learn how to use coding to draw

**Exercise 4.3: “Decision Tree”**

* Activity: Output balloons that float up through the viewing area, and that pop when the user gives a numeric input; larger number gives bigger pop
  1. Output balloons floating up through the screen without user input ⇒ compile ⇒ observe results
  2. Add user input, so that the balloon pops when the user gives a numeric input, with a bigger pop for a bigger number ⇒ compile ⇒ observe results
  3. Change shape type for balloons ⇒ compile ⇒ observe results
  4. Change color ⇒ compile ⇒ observe results
  5. Change size ⇒ compile ⇒ observe results
* Learning goals:
  1. Understand how to capture user input without blocking the code
  2. Learn how to create an animation that incorporates user intervention