


|  |   |   |
|--|---|---|
| <b>LESSON: Remix #1 (Mission 3-5)</b>  |   | <b>Time: 3 to 4 45-minute class periods</b>   |
| <b>Overview:</b><br><p>Students have completed three meaningful mission projects. Now is the time for them to master the skills and concepts in the missions by creating their own project. This is called a remix because students can use or reference previous code for their new remixed project.</p> <p>This project is an opportunity for students to be creative, collaborate with a partner and work on their teamwork skills and their coding skills.</p> |   | <b>Objectives:</b> <ul style="list-style-type: none"> <li>• I can summarize the projects from missions 3, 4, and 5</li> <li>• I can plan an program</li> <li>• I can create an original program based off of code from missions 3, 4 and 5</li> <li>• I can get feedback on my project</li> <li>• I can write an <b>if:else</b> conditional statement.</li> </ul> |
| <b>Standards:</b><br><p><b>2-CS-03</b> Systematically identify and fix problems with computing devices and their components</p> <p><b>2-AP-11</b> Create clearly named variables that represent different data types and perform operations on their values.</p> <p><b>2-AP-12</b> Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.</p>  | <b>CSP Framework:</b><br><p>Computational Thinking Practices:</p> <p>4.C Identify and correct errors in algorithms and programs, including error discovery through testing.</p> <p>6.A Collaborate in the development of solutions.</p>   | <b>Key Concepts:</b> <ul style="list-style-type: none"> <li>• <b>Code snippets</b> from previous programs can be reused and repurposed in a new project.</li> <li>• <b>Program development</b> follows a design process.</li> <li>• <b>Creating a new project</b> from the beginning is an excellent way for students to master their learning.</li> </ul>        |
| <b>Preparation:</b><br><p><b>Make a copy</b> of the assignment or put it in the LMS.</p> <p><b>Prepare</b> any formative assessments you want to use in the wrap-up</p> <p><b>Each student/pair</b> needs</p> <ul style="list-style-type: none"> <li>• a computer/ Chrome</li> <li>• a CodeX &amp; USB cable</li> <li>• assignment guide</li> </ul>  | <b>Links:</b> <ul style="list-style-type: none"> <li>• Codespace: <a href="https://sim.firialabs.com/">https://sim.firialabs.com/</a></li> <li>• <a href="#">Assignment document</a></li> <li>• <a href="#">Remix instructions slide deck</a></li> <li>• Daily reflection -- use your own link</li> </ul> | <b>Agenda:</b> <ul style="list-style-type: none"> <li>• Warm-up (5-10 minutes)</li> <li>• Remix plan (30-40 minutes)</li> <li>• Remix coding (40-80 minutes)</li> <li>• Peer review and coding (30-40 minutes)</li> <li>• Wrap-up &amp; Assessment (10 minutes)</li> </ul>  |
| <b>Vocabulary:</b> <ul style="list-style-type: none"> <li>• No new vocabulary during this lesson</li> </ul>  |   |   |
| <b>Assessment:</b> <ul style="list-style-type: none"> <li>• Daily reflection journal or Google form -- use your own link</li> <li>• Rubric (can be summative assessment) -- use "success criteria" checklist</li> </ul>  |   |   |


## Teaching Guide

### Warm-up (5-10 minutes)

 **Discuss** – Use a discussion strategy, like journaling, working at boards, selecting random students, or a form of think-pair-share.

Ask the students what they know about a remix. Go through slides 2, 3, and 4 in the [Remix Slide Deck](#)

### Activity – Remix Plan (40 minutes)

 Randomly group students into pairs for pair programming. The planning part can take a class period if they are following the process and taking it seriously and filling out the Steps #1-3 on the assignment document. Students shouldn't rush through the planning process.

#### Teaching tip – Step 1:

Go through slides 5 and 6.

Students should open their code for Missions 3, 4 and 5 (Pixels1, Display, & Music1). Then complete Step 1 in the answer document.

#### Teaching tip – Step 2:

Go through slides 7-10. Many suggestions for a remix are given. They can use one of them, or come up with their own. The purpose of giving suggestions is to help jog their brain into thinking about possibilities.

Students complete Step 2 in the answer document.

You can review the success criteria, or let students know there are program requirements, so they are aware when deciding on a project. The success criteria are at the bottom of the assignment document.

#### Teaching tip – Step 3:


Go through slides 11, and reference the Step 3 section of the assignment document. Planning is an important part of the programming process. Students may be reluctant to do a plan, but encourage them to do something, even if they change the program later or don't fully implement the plan. The idea is to get them started on the programming process so they will be comfortable using it as the programming difficulty increases, and also when they work on their program for the Create Performance Task.

### Activity – Remix Coding (40-80 minutes)

The length of time this takes will depend on the remix project students selected and the amount of planning they did. It could be one class period or two class periods.

#### Teaching tip – Step 4:

Go through slides 12-14.

 Important – use the sandbox for the coding (not a mission).  
left-hand corner above the toolbox.



The sandbox is located in the lower

 Important – start with a new file!!

Many good tips for coding are included. Students sometimes don't know where to begin. They just need to get started! They can use code from previous programs by either retyping or copying and pasting.

(optional): Review the [Mission Reminders slides](#).

Remind students that they should do a few lines of code at a time and test frequently. Also, they need to document their errors and how they fixed them. There is a table at the end of the document for this.

## Activity – Peer Feedback and improve project (30-40 minutes)

### Teaching tip – Step 5:

Go through slides 15-16.


Give students time to run at least one other group's program and give feedback. Use the table in the assignment document.


Students now have time to finish or improve their code, based on feedback. Also, students need to make sure their code is readable by including blank lines and at least two meaningful comments.

Students should review the success criteria and make sure their program meets all the requirements.

### Teaching tip – Extension:

If a pair programming team finishes extremely early, they are probably not challenging themselves. Encourage the team to add more functionality to their code. They shouldn't be satisfied with just meeting the requirements, but should create something they are proud of and that helps them practice as many skills and concepts as possible.

 Review the success criteria for completeness. Assignment is ready to turn in. Both students should include their names on the document.

 To turn in the assignment, students should download their code (FILE-DOWNLOAD), which will be a text file. Then they should submit their file through Google Classroom or your LMS.

*You may need to demonstrate this to your students*

## Wrap-Up (10 minutes)

Show slide 17. Wrap-up suggestion – have students do a gallery walk with each other's remix projects.

Formative Assessment:

- Daily reflection journal or Google form -- use your own link
- Programming journal (debugging chart only)
- Class discussion on what they learned from doing a remix
- Exit ticket

Summative Assessment: Use the success criteria to evaluate the remix project

### IMPORTANT!!

Students should clear their CodeX by running their "Clear" program.

If students haven't created a "clear" program yet, they can follow the steps in this [slide deck](#).



**SUCCESS CRITERIA:**

- Complete Step 1 on the assignment document
- Complete Step 2 on the assignment document
- Complete Step 3 on the assignment document
- Start with a new file and give your remix project a descriptive name
- Import modules (codex, time, etc.)
- Use at least one variable with a descriptive name
- Use at least one if statement
- Light up at least one pixel
- Display at least one image
- Play at least one audio file
- Display or print at least one text string
- Debug any errors in the code and keep a debugging table
- Get feedback on your program and make changes based on feedback (Step 5 on assignment document)
- Include a multiline comment at the top of your code that identifies its name and function
- Program is readable (blank lines)
- Program includes at least two meaningful comments (in addition to multiline comment at the top)