BRIGHT-CS

GAME TIME!

Software Development Project

Grade 6 | 5 – 7 Weeks

PROJECT OVERVIEW

FINAL PRODUCT
Students create and present a computer game using the Scratch development environment.

TIMEFRAME
5-7 weeks

AGE GROUP
Grades 6

DRIVING QUESTION
"How can we, as software developers, build a game?"

PROJECT DESCRIPTION
Why "Game Time"?

Software impacts just about everything we do; and its pervasiveness continues to increase every day. Students who understand how to write code are in a prime position to not only develop software solutions in the short term but to have a lucrative career in the long term. This module brings attention to key software development principles in the context of the Scratch Environment and encourages students to develop games that they would enjoy playing The overall goal of the module is to empower students with foundational Computer Science skills in a fun and engaging way.

HOW DOES THIS PROJECT DEVELOP INDEPENDENT LEARNERS IN STEM+CS?

All learners depend on adults at first. To persist in STEM+CS, they must "learn how to learn." Experiences that develop the habits of mind needed to take charge of one's own learning are less often provided to culturally and linguistically diverse students (Hammond, 2015). This project helps educators put all students on-course to becoming independent learners in STEM+CS.

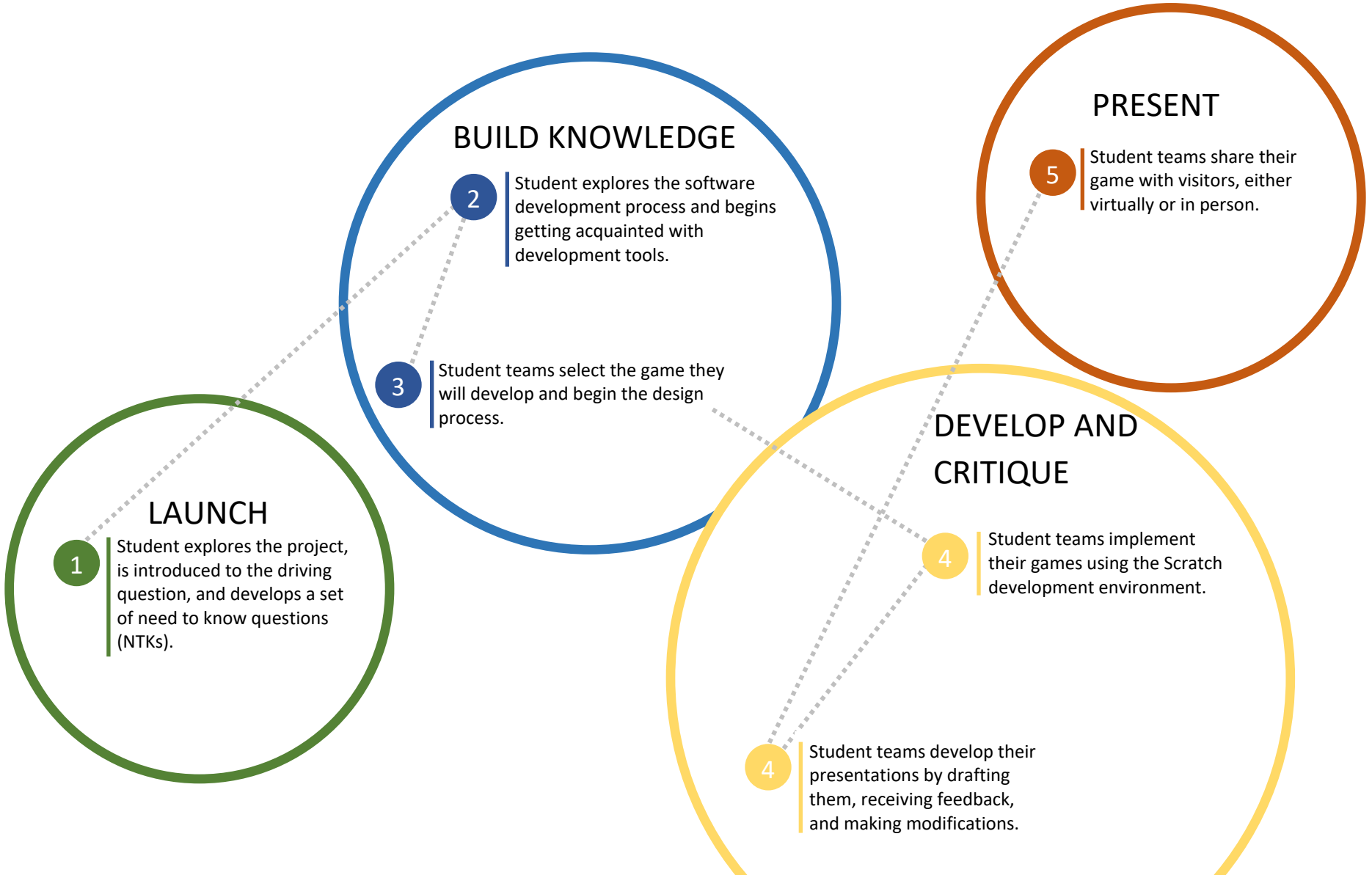
LEADS TO A MEANINGFUL PRODUCT
The first step to independence is engagement. Disconnected exercises that don't culminate in a real-life product are less likely to inspire thoughtful participation than lessons where students apply knowledge and skills to make something meaningful. The more likely they are to use the final product in their own lives, the more motivated they'll be to finish. And, when the product uses coding, they'll end up with authentic proof that "I can" do computer science. The resulting sense of efficacy can inspire even more engagement.

ALLOWS FOR PRODUCTIVE STRUGGLE
Independent learners have developed strategies for tackling new challenges (Hammond, 2015). How? They tell us that in STEM+CS, the process of trial and error is key (Yamaguchi et al., 2021a). Unlike mindless application of teacher procedures, struggling to figure out mistakes ("bugs") or make improvements themselves helps them not only see what strategies work, but really understand why. Of course, productive struggle takes time. A good project is paced to include multiple chances to make, reflect on, and resolve errors--from prototyping and debugging, to user-testing one's product and iterating, to practicing and honing a final presentation.

HELPS EDUCATORS CREATE CONDITIONS FOR SUCCESS
Struggling with challenges on its own is not enough to produce learning. Dependent learners are prone to drop back and let peers take over; even independent learners may shy from making valuable errors when others are watching (Yamaguchi et al., 2021a & 2021b). Educators create the conditions for success by providing the right inputs, feedback, and encouragement to help students embrace difficulties and break through. From building positive attitudes about mistakes, to types of prompts that let students get *themselves* unstuck, the "teaching tips" in this document help set up learners for mastering challenging tasks--the experience likeliest to make an impact on their future independence.

PROJECT PATH AND MILESTONES

The Game Time! project begins with a launch and ends when students formally reflect on their experience after presenting their final products. A bird's eye view of the project's path and milestones is provided below and visually summarizes how students will navigate from start to finish. A more in-depth presentation of student tasks follows on the next page.



PROJECT MILESTONES AND STEPPING STONES

Milestone #1: Student explores the project, is introduced to the driving question, and develops a set of need to know questions (NTKs).

ENTRY EVENT	PREVIEW THE PROJECT	EXPLORE THE DRIVING QUESTION
Student learns that software is an integral part of our lives, that someone has to develop that software, and that they can start actively building the skills they need to be a developer now.	Student is introduced to expectations for the final product: developing the concept for a game, designing and building the first level, and designing the second level.	Student examines different types of games, explores the question, "What does it take to build a game?", and develops a set of need to know questions (NTKs).

Milestone #2: Student explores the software development process and begins getting acquainted with development tools.

WHAT IS SOFTWARE DEVELOPMENT?	GETTING ACQUAINTED WITH SCRATCH
Student rounds out the NTKs and leverages them to discuss baseline software development knowledge and skills.	Student gets acquainted with the Scratch development environment in a context where the instructor makes a clear connection between the NTKs and software development.

Milestone #3: Student teams select the game they will develop and begin the design process.

IDEATE AND EVALUATE	DEVELOP WIREFRAMES
Student teams brainstorm the type of game they want to develop and the objective of the game.	Student develops wireframes individually and then collaborates with a teammate to merge the best aspects of each design.

Milestone #4: Student teams implement their games using the Scratch development environment.

DEVELOP THE PROTOTYPE	IMPROVE THE PROTOTYPE	IDEATE LEVEL 2
Student teams work together to implement the first level of their video game in the Scratch environment.	Team members rotate to review games and provide feedback. Teams then make improvements based on input.	Student teams build wireframes for Level 2 and start generating a list of to-dos and NTKs to implement the next level.

Milestone #4: Student teams develop their presentations by drafting them, receiving feedback, and making modifications.

DRAFT AND PRACTICE	INCORPORATE FEEDBACK
Student teams draft their presentations and practice in front of other teams.	Student teams improve their presentations based on feedback received from their peers and present to visitors to gain more feedback.

Milestone #5: Student teams share their game with visitors, either virtually or in person.

PREPARE	PRESENT	REFLECT
Student teams incorporate feedback received from visitors and make final preparations for presenting their projects.	Student teams present their games to experts either virtually or in person.	Student and teacher reflect on their original NTKs, discuss their current understandings, and identify opportunities to extend learning.

LAUNCH – introduce students to the project and get them excited about it.

BUILD KNOWLEDGE – provide the baseline knowledge, activate the baseline skill necessary to complete the project, and allow students to build their plan.

DEVELOP AND CRITIQUE – allow students to implement their plan and provide opportunities for them to receive feedback from their peers.

PRESENT – allow students to develop and deliver presentations for an audience outside of their teacher/peers.

GETTING STARTED

Where

This project is appropriate for any setting, from a core STEM class, to an elective course in computing, to a homeroom, extracurricular club, or camp. Educators do not need to be experts in computing to lead it. All of the content knowledge and skills necessary to complete the project are outlined in linked PowerPoint presentations and other resources that are freely available on the web. We do, however, recommend that educators take time to familiarize themselves with these resources, and even attempt to complete the project on their own first. The experience of creating one's own computing product can provide valuable insight for supporting students through the process.

Who

Student recruitment is an important consideration for educators seeking to engage diverse students in this project, particularly when planning to teach it as part of an elective course, club, camp, or other setting outside a core STEM class. Relying on interested students to identify themselves may result in missing out on some who could develop an interest if only for the right kinds of outreach. Effective recruitment methods include: 1) asking influential adults (such as parents of, and school staff popular among, culturally and linguistically diverse students) to personally invite students to participate; 2) organizing recruitment sessions to show students a finished exemplar of the final product they will create; and 3) asking attendees to bring a friend to the sessions with them.

It's also worth considering how to ensure a diversity of prior CS skills and experiences among your students. Research suggests that among culturally and linguistically diverse students, both dependent and independent learners find it helpful to work with like peers at different levels of skill (Yamaguchi et al., 2021b). That's because peers with more skill allow students to seek support and experience perspectives on the material beyond what their teacher can provide alone; peers with less skill offer students the opportunity to build confidence by explaining the material to others. Since there may be few culturally/linguistically diverse students in any given grade level, and particularly few diverse students of the same gender, it could be worth recruiting across different grade levels--even different schools--to create a cohort where diverse students can find a number of peers like them who also bring different skills and experiences to the table.

When and What

After recruiting a cohort of diverse students, survey them to see which potential meeting times work best (assuming you are teaching the project as part of an extracurricular club or camp). Then secure the necessary hardware. While all of the software recommended for this project is freely available on the web, students will need access to individual, internet-enabled devices (preferably laptops) during each session of the class, club, or camp. Consider what existing devices may already be available to students, whether through school issue, computer labs, local public libraries, or other facilities.

END STATE

More than completing a final product, this project is about helping students develop the academic, cognitive, and social-emotional tools they need to take charge of and propel their own learning in STEM+CS. But what does that really look like? How can you tell which students haven't yet had experiences to develop those tools, in order to better support them? And what does it look like when you've done your job to coach them towards greater independence?

BASED ON RESEARCH (Hammond, 2015; Yamaguchi et al., 2021a):

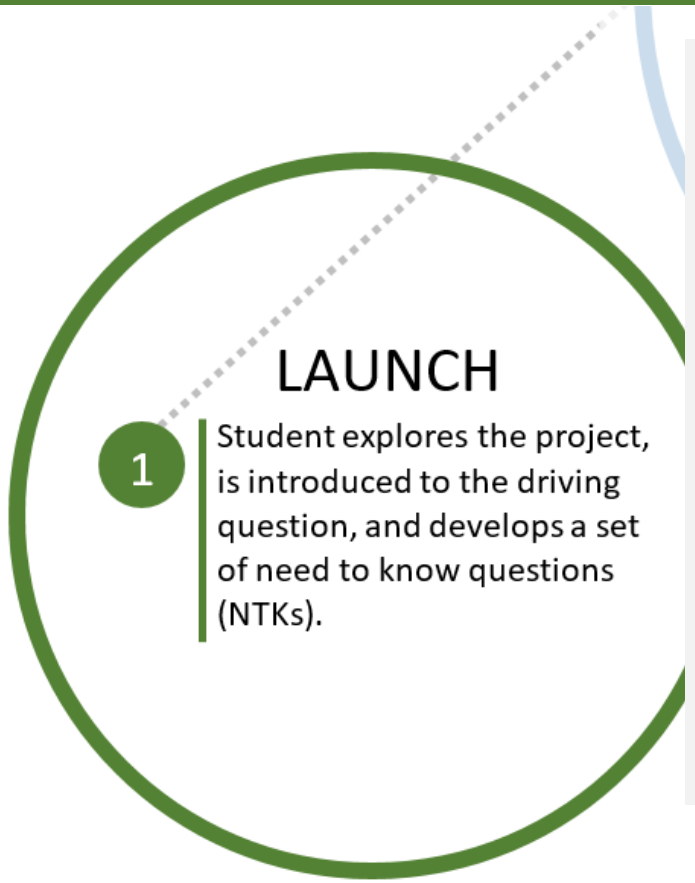
Dependent learners...		**Independent learners...**
• **Often think mastery means being able to name and remember procedures for completing a task** (i.e., rote learning) • **May engage when asked to reproduce teacher-modeled skills/knowledge with minimal cognitive effort** (e.g., answering simple recall questions, copying a teacher's steps) • **Stop engaging when challenged to "figure out" new content themselves** Zaretta Hammond (2015) calls this "depend[ing] on the teacher to carry most of the cognitive load of a task."	***Acquiring new skills and knowledge***	• **Usually appreciate that mastery means knowing *why or how* "it works"** (i.e., conceptual understanding) • **Often prefer to jump in and "figure out" new skills/knowledge after minimal instruction** (e.g., through trial and error). Hammond (2015) calls this "rel[ying] on teacher to carry some of the cognitive load temporarily."
• **Tend to engage less on aspects of their project that challenge them to develop new skills in STEM+CS** (e.g., coding) **than in aspects where they already have skill** (e.g., visual presentation) • **When working in groups, sometimes avoid performing tasks requiring new STEM+CS skills**, allowing others to carry the cognitive load • **Sometimes declare their work "finished" before it reaches their original goal**, when further improvements would require pursuing skills or knowledge beyond those provided by teacher	***Managing independent work time***	• **Tend to spend more time on aspects of their project that require new skills in STEM+CS** • **Are likelier to continue working on their products when there are still improvements to be made,** even if those improvements require pursuing skills or knowledge beyond those provided by teacher
• **When running into challenges in their work, are more likely to get "unstuck" by receiving help from a teacher** than to use any other strategies	***Getting "unstuck"***	• **Are less likely than dependent learners to ask their teacher for help and more likely to try other strategies for getting "unstuck,"** such as creating a plan of attack, rereading materials provided, conducting their own research, tackling easy parts first, or using trial and error

MILESTONE #1: LAUNCHING INQUIRY

As the entryway into Game Time!, this milestone is designed to fuel the interests, excitement, and creative curiosities of students and provide the support they need to be ready for and invested in the process ahead of them. Specifically in Milestone #1, students begin having foundational discussions and participating in activities that will set them on a path to answer the driving question:

"What does it take to design and develop a video game?"

Milestone #1 ignites an inquiry cycle that will ultimately be shaped by students' own creative curiosities and directed by an evolving list of need to know (NTK) questions that they must define and address in order to make progress. Since NTKs can help students be more mindful about the concept or ideas they will explore, it is important to encourage this process and help students to see the value in it. This might include anticipating students' questions in advance and structuring the lesson accordingly. It must also mean being flexible to NTKs as they arise and ensuring that students have a safe space to tap into and explore their curiosity. For this project, Game Time!, students' NTKs may relate to the role of a software developer, how software is developed, the kinds of games they can develop, and the ideation process in this context. Although you may be able to predict a large majority of NTKs in advance, students will likely venture into unanticipated territory, generating questions that are unique, unexpected, and novel. Helping students value this creative curiosity and understand how to channel it would be a significant outcome of this project.



STEPPING STONES

ENTRY EVENT

Student learns that software is an integral part of our lives, that someone has to develop that software, and that they can start actively building the skills they need to be a developer now.

PREVIEW THE PROJECT

Student is introduced to the expectations for the final product: developing the concept for a game, designing and building the first level, and designing the second level.

THE DRIVING QUESTION

Student begins exploring the question, "What does it take to build a video game?" and develops a list of need to know questions (NTKs) to guide their inquiry about the software development process in general and the game development process in particular.

RESOURCES

PROJECT SUPPORT

- Slide Deck: Milestone 1

ARTICLES

- Occupational Outlook - Software Engineers
- Skills Needed For Software Development
- Careers in Game Design

VIDEOS

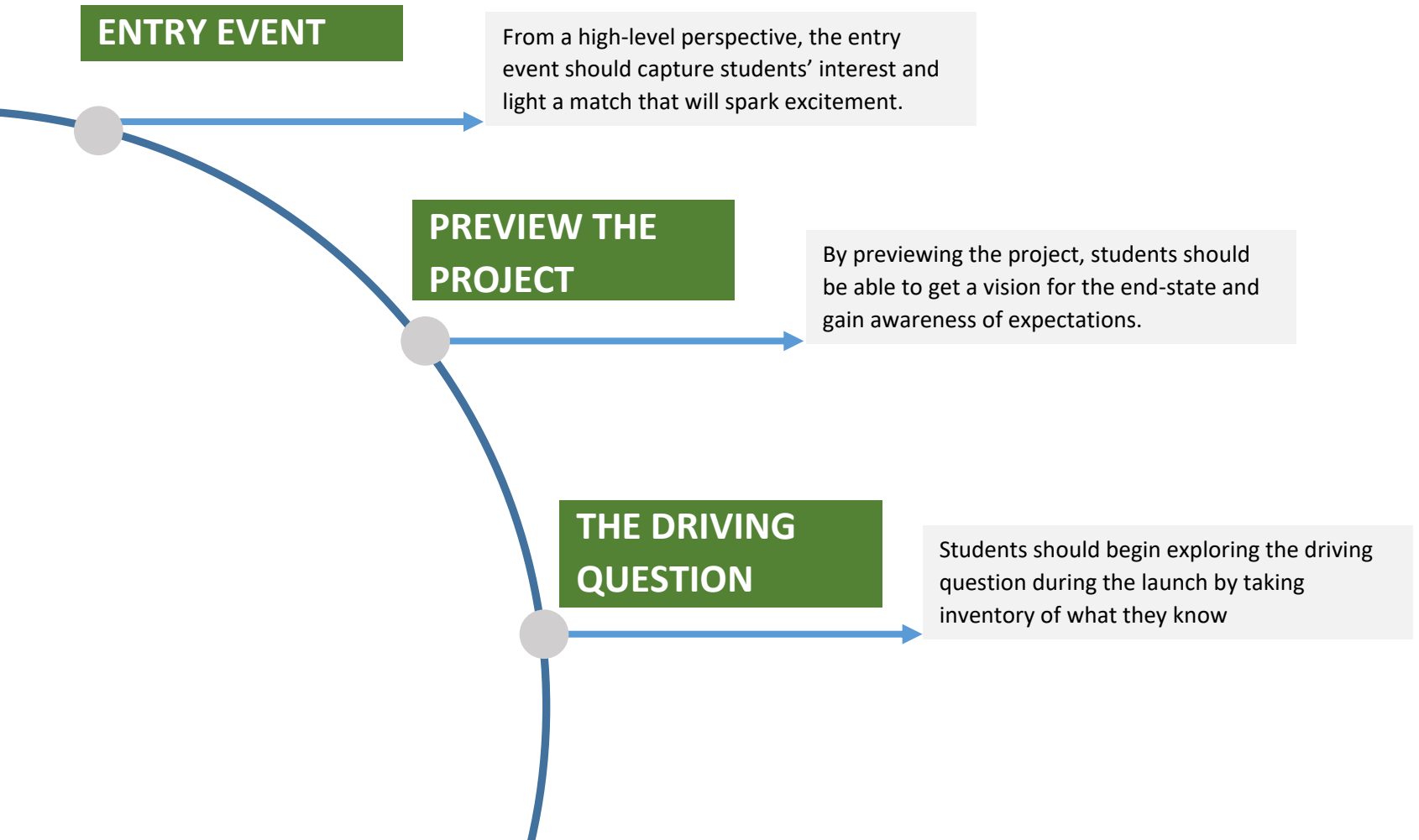
- 5 Reasons Why I Love Being a Software Engineer
- A Day In the Life of a Software Engineer
- What Do I Do as a Software Engineer
- What Is Software Development Different Kinds of Video Games

ACTIVITIES

- Student Planning Sheet
- Project Rubric
- Anticipation Guide
- Develop Critical Thinking Questions

DESIGNING AND EXECUTING THE LAUNCH

Designed and executed carefully, Milestone #1 provides an opportunity to pique students' curiosity, encourage discussion, and help participants start building confidence that they are capable of success in open-ended problem solving in general and data science in particular. The accompanying slide deck provides one way to execute the launch; the text below provides more context about what each component of the launch is to accomplish. In general, the key is to introduce and emphasize the decision-making, planning, and end-product development process in an engaging way. The remainder of this document provides more detailed insight into the launch components, desired outcomes, and insight into how you might implement each *component* to achieve critical *outcomes*.



NEED TO KNOW (NTK) PREPERATION

By developing, exploring, and tracking their NTKs during this project, students will not only be more mindful of their curiosities but they will also see how getting further in the process will resolve some questions and spark new ones. Understanding how to navigate this process is a valuable skill and nurturing this skill will be helpful to students in this project and beyond.

The NTK process is essential to the project. It helps students to activate their prior knowledge and to be mindful of the questions they have as they explore the driving question. Given the nature of this project, students are likely to have high-level NTKs about software development and game development in Milestone #1. Potential questions are listed below.

GENERAL SOFTWARE DEVELOPMENT QUESTIONS

- What do software developers/engineers do?
- What is the formal definition of software?
- Is software development 'hard' and if so how?
- How long does it take to create software?
- What skills do you need to be a software developer?
- What can you do to develop software development skills (e.g., training, education, etc.)?

GAME DEVELOPMENT QUESTIONS SPECIFIC TO THIS PROJECT

- What are the different kinds or types of games?
- What types of games do I like to play and why?
- How hard is it to develop a game?
- How do I start a game development project?
- What coding language will I use?
- What makes one coding language better than another?

SAMPLE LAUNCH: STUDENT VIEW

LESSON LAUNCH

THE ENTRY EVENT

Software helps us do just about everything! From writing papers (Microsoft Word) to developing presentations (PowerPoint), to engaging with our friends (Facebook) and sharing fun videos (TikTok), we use software just about all day long to accomplish a wide variety of tasks. Even the cars, Metro buses, rail, and Uber that help us travel and get around rely on software.

This module is designed to dig into the software engineering process with a special focus on the development of a game!

Quick Tip!

From the start of your first lesson, students will begin assessing their interest level in what you'll teach and the work you'll ask them to do. This often influences how much effort they put in later, or even--if the experience is optional (as with an afterschool or summer program)--whether they return for the next session. So it's important to consider how you'll ensure that students leave Milestone 1 excited to come back and roll up their sleeves.

INVESTIGATE

PREVIEW PROJECT EXPECTATIONS

Your project is to design a game and implement it. Over the course of the project, you will become familiar with the game development process. Your final product will be (1) the first level of your game, (2) the design for the second level, and (3) a presentation about both your game and your development process. The audience for your final presentation will be a group of visitors. However, you will present to your peers in advance to get practice and feedback.

After exploring the specifications and expectations for the project (Student Planning Sheet, Rubric) you will complete an activity based on the Anticipation Guide to start thinking about what you would like to learn/accomplish as a result of this project.

Build interest in the final product

The more meaningful the final product to students, the more likely they are to stay engaged. For both dependent *and* independent learners, a meaningful product is one that feels relevant to their lives. Do they have a personal use for it? As early as possible, make sure not only that students are clear on what their final product could look like but also that they've begun to imagine how the product could be useful to them right here where they live, right now at this age.

INVESTIGATE (CONT'D)

EXPLORING THE DRIVING QUESTION

"What does it take to design and develop a video game?"

INITIATING THE NEED TO KNOW PROCESS

Drawing from the discussions you've just had, let's organize our shared knowledge. What do you already know about software development and video games? What do you need to learn in order to be able to develop your own games? What questions do you have about what you need to have or know in order to develop your game?

Additionally, people who implement computer-based games professionally are called software developers or engineers. What do you know about what it means to be a software engineer? Do you know any software engineers? What do you think you need to know or learn in order to a "real" software developer?

SYNTHESIZE AND REFLECT

CLOSING THE LAUNCH

Review the Student Planning Sheet and use the appropriate space to write down each of the key decisions you and your partner made about developing your game and also keep track of alternatives (so you can return to them later).

Once you and your partner have discussed your options and identified a preliminary plan, provide your instructor with the following:

- The make-up of your team (student names)
- The game you plan to develop

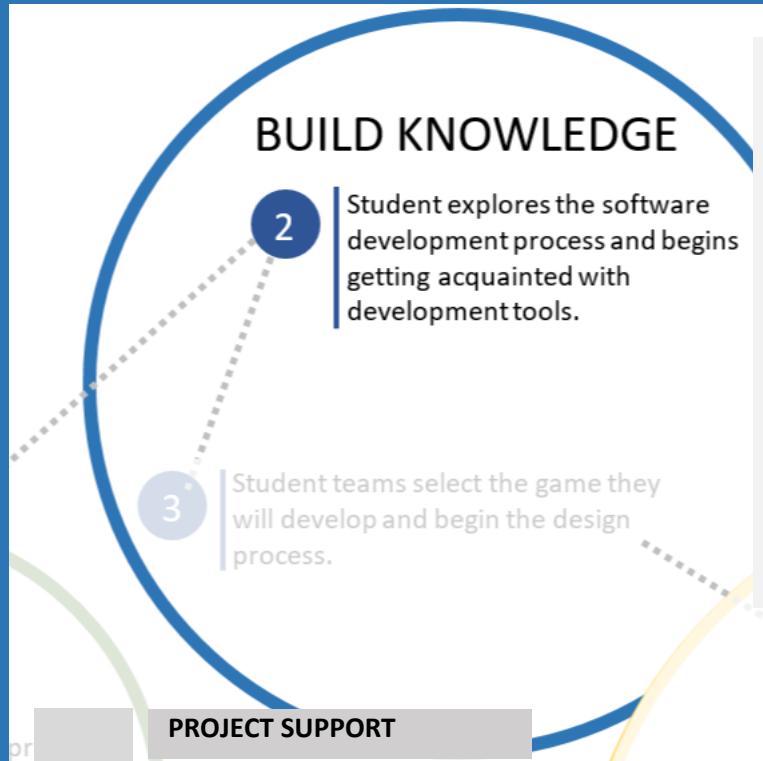
Pair students with intention and build relationships

The minute students learn they'll be working in pairs, they start worrying about who their partner will be. In STEM especially, it's important for girls and culturally/linguistically diverse students to have the chance to work alongside others like them, which may reduce stressors like stereotype threat and more. This could mean allowing students to choose partners who they already know, but it could also mean assigning pairs and giving tasks that build trust between partners early in the process. Positive working relationships with others in the class can not only support students to begin taking academic risks in front of their peers, but can also become an important reason for students to keep showing up to class.

MILESTONE #2: SOFTWARE DEVELOPMENT INTRODUCTION

This milestone immerses students in the role of a software developer, helps students explore key skills/approaches they will need in the rest of the project, and aims to expand awareness of software development - as a career - in a way that is both tangible and appealing. In addition, it provides baseline exposure to the Scratch programming environment and allows students to be guided through the debugging process so that they (1) can get familiar with the kinds of errors they will encounter and (2) feel more confident addressing them in the future. The goal of the latter exercise is to not only help students get a deep dive into the Scratch environment but to also help them feel more comfortable trying new things (without being afraid of breaking the system).

Understanding basic ideas about sprites (graphics and characters), variables (to keep score), loops, and event listeners (that recognize when characters, things, or both move, collide, or change status in some other way) are key to game development. The goal of this milestone is to help students learn and apply each in a hands-on way that allows them to build the appropriate knowledge, make mistakes, ask questions, gain exposure/familiarity, and ultimately be prepared to develop systems independently (in both the short- and long-term).



STEPPING STONES

WHAT IS SOFTWARE DEVELOPMENT?

Students round out the NTKs and leverage them to discuss baseline software development knowledge and skills.

GETTING ACQUAINTED WITH SCRATCH

Student gets acquainted with the Scratch development environment in a context where the instructor makes a clear connection between the NTKs and software

RESOURCES

PROJECT SUPPORT

- Slides: What is Software Development
- Slides: Getting Acquainted with Scratch

VIDEOS

- Scratch Coding Overview for Beginners
- How to Debug Your Code in 4 Simple Steps!
- Debugging Code in Scratch Activity
- What is Scratch
- Why Use Scratch
- Introduction to Scratch Programming
- Game Development Basics
- Algorithms and Debugging Using Scratch

ARTICLES

- Scratch for Educators
- Introduction to Programming in Scratch
- Debugging Scripts
- Top 5 Errors In Scratch Programs
- Common Bugs in Scratch Programs

ACTIVITIES

- Jot Thoughts
- Generative Summarizing
- Ask Your Authentic Guiding Questions
- Develop Critical Thinking Questions
- KWL Chart

Build students' muscle for independent research

Independent learners tell us that one strategy they use for tackling challenges in their work is to conduct research. But to track down the knowledge or skills they need, students must learn to articulate what their specific questions are (*What now?* doesn't count!), then practice searching out answers among the endless resources available online. Give students a chance to flex this muscle by offering the linked texts and videos as a starting place to explore their NTKs. Then, in the likely case they still have unanswered questions, encourage them to go further by asking your own questions to spur reasoning about what to do next (e.g., "What search terms might you try?"). This can produce greater independence in the long run than simply telling them what steps you'd take or, worse, what the answers are.

	1. WHAT IS SOFTWARE DEVELOPMENT?
Standards	CCSS.ELA-LITERACY.SL6.1.C; CCSS.ELA-LITERACY.SL6.1.D; CCSS.ELA-LITERACY.SL6.2
Accompanying Slides	Slides: What is Software Development
Students will be able to....	Expand/address their list of NTKs by discussing basic ideas about software development (both as a process and a career) and identifying key factors students will need to either (1) define or (2) get more information about in order to move forward.
Ideas for activities	• Help students understand the world of software development and provide the baseline, programming language-independent knowledge necessary to conceptualize the software development process via an interactive presentation that encourages group discussion via questions. The discussion should begin with a review of the NTKs (to keep them front of mind) and focus on the following: ○ **Creative curiosity** and NTKs are important as students navigate the project. ○ Software development is a combination of problem solving and language translation. ○ Much like natural (human) languages, there are different kinds of languages used for software development. ○ Software developers use a variety of skills; discussing these skills can help students prepare for the project and discussing their earning potential can help to engender interest in the profession. Towards the end of this discussion, review the NTKs and specifically identify which ones have been covered to date and how. • **Suggestion**: have students keep track of new words and concepts they learn during this process for later reflection.
Reflection and Synthesis Prompts	• What aspects of software development are interesting to you? • What responsibilities and tasks of a software developer fit your skills and interests? Why? • What factors of the software development process seem to be the most difficult?

Formative Assessment Ideas	● Have students explain the relationship between natural and computer languages. ● Invite students to explain what a software developer does and what skills they need.
Suggestions for Feedback and Support	● Have students discuss the new keywords, phrases, and ideas they've learned as a result of the "WHAT IS SOFTWARE DEVELOPMENT?" module. ● Have students explain, in their own words, what they think creative curiosity is, why it's important, and whether they have used it in other contexts. ● Have students discuss how they feel about working on an open-ended project, where the answers are not known in advance.

Resist carrying all of the cognitive burden for students

As educators, it's tempting to break new skills into steps we can model while students follow along. But this can lead students to reproduce our steps passively with little mental effort, limiting their understanding and eliminating a chance to develop habits of mind for tackling new challenges independently. Instead, explain the Scratch concepts in the slide deck via a sample program as students watch--*without* letting them copy you. Run through only as many concepts as you can in five minutes or less, then set students loose in pairs to try for themselves, with a copy of your slides for reference. Repeat as many times as needed to introduce all the concepts. The key is to get students experimenting with the concepts as quickly as possible, showing just enough to spark their thinking – and no more.

	2. GETTING ACQUAINTED WITH SCRATCH
Standards	CCSS.ELA-LITERACY.SL6.1.C; CCSS.ELA-LITERACY.SL6.1.D; CCSS.ELA-LITERACY.SL6.2; CCSS.ELA-LITERACY.RST.6-8.3
Accompanying Slides and Key Videos	• Slides: Getting Acquainted with Scratch • Scratch Coding Overview for Beginners • How to Debug Your Code in 4 Simple Steps! • Debugging Code in Scratch Activity
Students will be able to....	Continue to actively resolve NTKs and apply baseline software development knowledge in the Scratch development environment. In effect, Scratch will be used to apply the knowledge gained in the "**WHAT IS SOFTWARE DEVELOPMENT?**" module in an actual development environment.

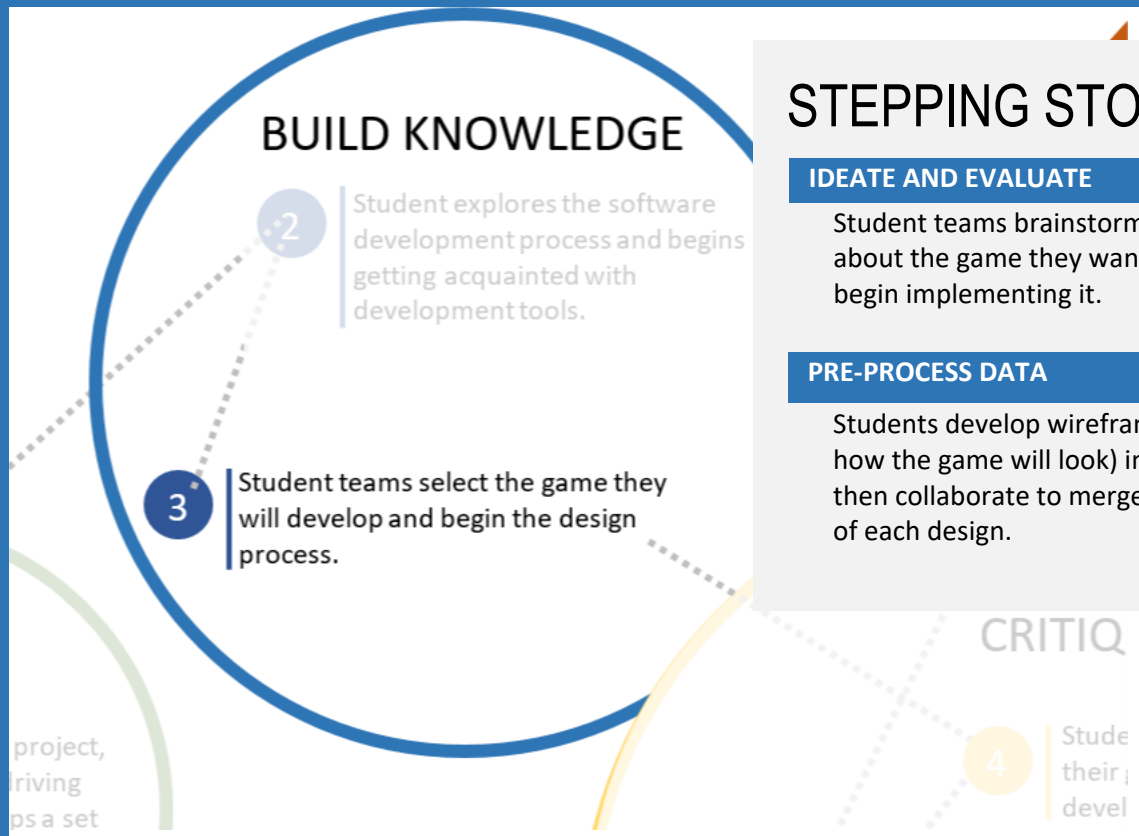
Ideas for activities	• Actively/iteratively discuss and apply key Scratch concepts by having students start the environment, begin a sample project, and walk-through its execution as shown in the Scratch Coding Overview for Beginners video. Use the context of the "**WHAT IS SOFTWARE DEVELOPMENT?**" module to introduce this topic as follows:
	  ○ Discuss that Scratch is a kind of programming language and compare to natural languages.
	  ○ Discuss why syntax has a different meaning with Scratch since it is graphic language.
	• Show students a sample program and have students develop a concept map between the way the program operates and the underlying code.
	  ○ The goal here is to reinforce that:
	  ▪ there is a link between an English algorithm and the code that builds the program.
	  ▪ Syntax is important; the reason the code 'works' is because it follows specific rules.
	• Review the following videos with students in the order shown to (1) introduce the idea of debugging and (2) explore/correct 'buggy' programs:
	  ○ How to Debug Your Code in 4 Simple Steps!
	  ○ Debugging Code in Scratch Activity
	• In advance of this session, create a Scratch program and introduce errors into it. During this session, have students load the program you developed and allow them to find the bugs in the program independently:

	○ This is an important opportunity to drive home the discussion of syntax as it relates to software development and errors. ● Show students examples of the kinds of games that will be developed: maze, catch, and whack-a-mole. Have students revisit their original game decision and decide whether they want to stay with their original choice, or make a change.
Reflection and Synthesis Prompts	● What factors about Scratch are hard to grasp? ● What are your impressions of the Scratch environment? Is it easy to use? Hard? ● Does working in the Scratch environment change your opinion of the kind of game you want to develop? If so, why?
Formative Assessment Ideas	● Have students identify the code given an algorithm. ● Have students identify the algorithm, in English, given the code. ● Have students review code with improper syntax, identify the errors are, and discuss how to fix them.
Suggestions for Feedback and Support	● Make sure students are following the lesson and understand the important concepts they'll need in order to develop their game. ● Have students revisit their NTKs at this juncture, discuss what they have recently been able to answer, and what they still need to know. ● Lever the KWL worksheet to further review what has been learned so far and figure out how it can be applied toward the project.

MILESTONE #3: IDEATION AND WIREFRAMING

This milestone immerses students in the brainstorming process for game development.

Critical steps in this milestone include identifying the object, look and feel, and structure of the game. Given that this is a team activity, collaboration, ideation, and sketching are inherent parts of this milestone as well, and will allow students to exercise and strengthen soft skills in the process.



STEPPING STONES

IDEATE AND EVALUATE

Student teams brainstorm specific aspects about the game they want to develop and begin implementing it.

PRE-PROCESS DATA

Students develop wireframes (sketches of how the game will look) individually and then collaborate to merge the best aspects of each design.

RESOURCES

PROJECT SUPPORT

- Project Specifications

VIDEOS

- The Need to Wireframe
- A Better Way to Brainstorm: How to Get Students to Generate Original Ideas
- Brainstorming Techniques How to Innovate in Groups

ACTIVITIES

- Think-Pair-Share
- Exit Tickets
- Points of Most Significance
- Two Stars and a Wish

STEPPING STONES TO IDEATION AND WIREFRAMING

	1. **IDEATE AND EVALUATE**
Standards	CCSS.ELA-LITERACY.WHST.6-8.7; CCSS.ELA-LITERACY.RST.6-8.3; CCSS.ELA-LITERACY.RST.6-8.7; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.CCRA.W.7; CCSS.ELA-LITERACY.CCRA.W.8; CCSS.ELA-LITERACY.CCRA.W.9; MS-ETS1-1; MS-ETS1-2
Students will be able to....	Understand the importance of prototyping game designs and develop their first wireframe for a game they like/play.
Ideas for activities	● Introduce the idea of wireframes for planning/designing a game, show the students an example wireframe, and then have students develop a wireframe for a game they like and/or play.
Reflection and Synthesis Prompts	● What value do you think wireframing adds to the game development process? ● Why do you think it may be helpful to wireframe your game before you begin development? ● How can wireframing help an individual and a team finalize their game design?
Formative Assessment Ideas	● Have students share: ○ the wireframes developed for a game they like/play with you and their teammate, ○ their process for developing the wireframe and any difficulty they had, and ○ their perspective for why wireframing is critical.

Pose students' questions and problems back to the class

As students begin tackling more technical aspects of their project, it's likely they'll have questions or run into problems. A key skill of independent learners is being able to get themselves "unstuck" using strategies besides just asking the teacher how. Take this opportunity to add to students' strategies for getting unstuck, by challenging the whole class to reason through ways to resolve individuals' questions or problems. As students discuss, make observations about the problem or ask your own questions to prompt student thinking about the underlying concepts. This approach also has the added benefit of helping to normalize experiences of difficulty as a regular part of the learning process.

STEPPING STONES TO IDEATION AND WIREFRAMING

Set expectations for partner work

When partners bring different levels of independence to their work, it's common for more independent learners to take over the cognitive burden, while dependent ones sit back or focus on aspects of the project that don't require them to practice the new skills it seeks to teach. Now is the time not only to set expectations that partners share the work equally, with each student getting the chance to attempt new STEM skills (particularly coding) individually--but also to emphasize why it's important. Use the Project Rubric provided in Milestone 1 to lead a class discussion where students develop "pair work norms" to ensure each partner gets to demonstrate all of the competencies outlined in the rubric.

	2. DEVELOP WIREFRAMES
Standards	CCSS.ELA-LITERACY.WHST.6-8.7; CCSS.ELA-LITERACY.RST.6-8.3; CCSS.ELA-LITERACY.RST.6-8.7; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.CCRA.W.7; CCSS.ELA-LITERACY.CCRA.W.8; CCSS.ELA-LITERACY.CCRA.W.9; MS-ETS1-3; MS-ETS1-4
Students will be able to....	"Plot" their game design on paper (via wireframing), refine their design based on feedback from the instructor, and work with their team members to combine their best ideas into a final wireframe. Individual and, ultimately, collaborative wireframing will give students multiple opportunities to practice this skill.

Ideas for activities	• Provide students with the project specifications and reminder about the kinds of games they can develop before they begin working on their wireframes.
	• Have students, separately, develop the wireframe for the game their team has agreed to develop, share with you, and then refine based on your feedback. Next, have students swap designs, and create a new wireframe that captures the best aspects of each design.
	• Have students present the following to the class:
	  ○ The type of game they plan to develop
	  ○ The objective of their game
	  ○ The final wireframe

Reflection and Synthesis Prompts	• How much did your idea about how the game should look change when you collaborated with your team mate? How much did it stay the same? • How did you decide which frames were key to include in your wireframe? • Take a look at your NTKs. How many have you been able to answer so far? Do you need to do more research to round out any of the answers you have so far? Of the NTKs that have not been answered, when do you think you will be able to answer them? Are any of the NTKs beyond the scope of this project?
Formative Assessment Ideas	• Review the wireframes developed so far and assess how well they capture the design for the game. • Have students submit an overview of the skills they think they've learned since the start of the project and those they think they've improved since the beginning of the project. • Have students also anticipate what other skills they may learn/strengthen in the rest of the project.

MILESTONE #4: GAME DEVELOPMENT

The goal of this milestone is to have students apply the software development skills they developed/refined in Milestone #2 and the wireframe they generated at the end of Milestone #3 toward development of a video game via pair programming in the Scratch environment.

STEPPING STONES

DEVELOP THE PROTOTYPE

Student teams work together to implement the first level of their video game in the Scratch environment.

IMPROVE THE PROTOTYPE

Team members rotate to review games and provide feedback. Teams then make improvements based on responses.

IDEATE LEVEL 2

Students build wireframes for Level 2 and start generating a list of to-dos and NTKs for implementing the next level.

RESOURCES

ARTICLES

- Pair Programming Guide
- Benefits of Pair Programming
- Pair Programming in the Classroom

VIDEOS

- Pair Programming in Action

ACTIVITIES

- Know-Need To Know
- Anticipation Guide
- Plus, Minus, Interesting
- Critique Protocol

Provide strategic support so students feel a sense of mastery

Experiences of success in overcoming challenges, or "mastery experiences," are critical to developing confidence--a key quality of independent learners. But students may attribute success to their teacher's help rather than their own efforts, if given the wrong types of support. At this stage of the project, students are especially likely to run into challenges (like bugs in their code), so it's important for teachers to provide support that preserves students' chance to experience personal success. When they ask you for help with these challenges, resist the urge to give them solutions. Instead--along with questions or observations to prompt their thinking as described in Milestone 3 (Pose students' questions and problems back to class)--provide encouragement that "you can do it," based on specific evidence of the student's past success (for example, as they explore and resolve common Scratch errors in Milestone 2). In the end, they will be more likely to feel they resolved their challenges "themselves"--and can do so again

	1. DEVELOP THE PROTOTYPE
Standards	CCSS.ELA-LITERACY.WHST.6-8.7; CCSS.ELA-LITERACY.RST.6-8.3; CCSS.ELA-LITERACY.RST.6-8.7; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.CCRA.W.7; CCSS.ELA-LITERACY.CCRA.W.8; CCSS.ELA-LITERACY.CCRA.W.9; MS-ETS1-1; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4
Students will be able to....	Implement the first level of the game in Scratch using a pair programming approach.
Ideas for activities	• Present the pair programming model to students and allow them to build the first level of their game using this approach. • Have students create to-do lists or checklists to help them organize and keep track of their development process.
Reflection and Synthesis Prompts	• What are the pros and cons of the pair programming approach? • Were there any decisions you made about your game that were hard to implement? If so, did you create a work-around or power through? • What did your group accomplish today? What will you need to do tomorrow (or the next time we meet)?
Formative Assessment Ideas	Students can share, orally or in writing, their goals for each work day, what they accomplished, and next steps. • Writing goals, accomplishments, and plans down will help students compare what they planned to accomplish to what they actually accomplished and, somewhat quantitatively, keep track of their progress • Keeping written notes of their planning and implementation processes will likely be very helpful to students during reflection, at the end of the project, as well.

Make it normal to have errors and misunderstandings

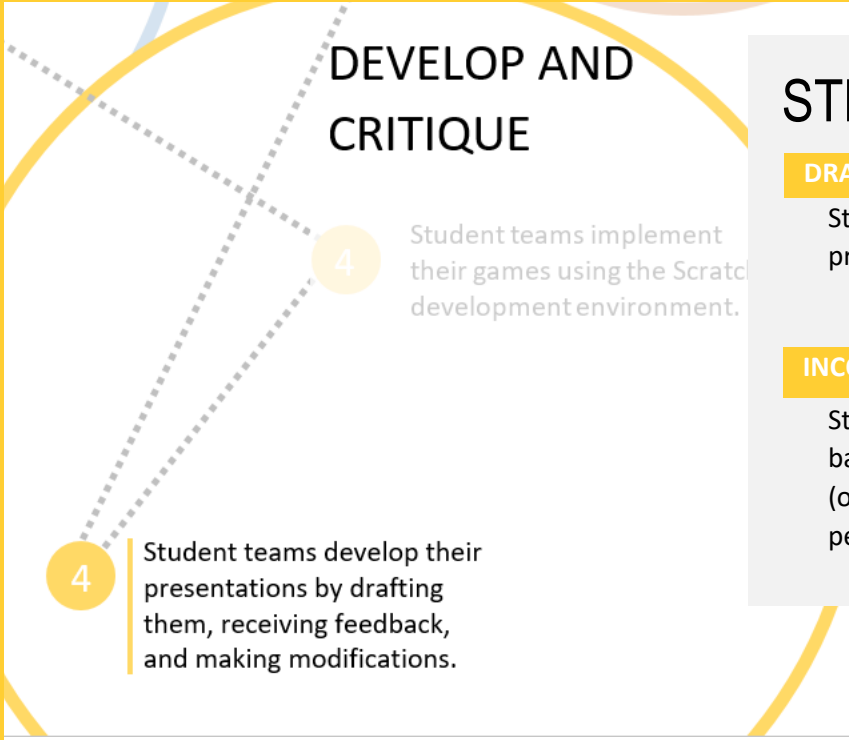
Both dependent and independent learners can be reluctant to make errors or reveal misunderstandings, especially in front of their peers. But the success of peer critiques hinges on their willingness to do so. It's your task to make "presenters" comfortable talking about their team's work--*especially* any problems they still have--and to make "rotators" comfortable asking questions of other teams without worrying about "sounding dumb." After explaining the roles of presenter and rotator, try reducing students' anxiety about making their mistakes public by talking about an experience in school or at work where *you* made mistakes. Explain how you resolved them, highlighting any ways you solicited feedback from others or asked people questions in the process. Encourage students to see mistakes as part of learning, and their peers as potential resources to help resolve their errors and misunderstandings while they still have time before final grades are assigned.

	2. IMPROVE THE PROTOTYPE
Standards	CCSS.ELA-LITERACY.SL6.1.C; CCSS.ELA-LITERACY.SL6.1.D; CCSS.ELA-LITERACY.WHST.6-8.7; CCSS.ELA-LITERACY.RST.6-8.3; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.SL.6.4; CCSS.ELA-LITERACY.CCRA.W.7; CCSS.ELA-LITERACY.CCRA.W.9; CCSS.ELA-LITERACY.CCRA.SL.1; CCSS.ELA-LITERACY.CCRA.SL.2; MS-ETS1-1; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4
Students will be able to....	• Review the level 1 game implementation of their classmates. • Provide critiques to other team members about how to improve their games, field critiques from other teams, and modify their implementation accordingly.
Ideas for activities	• Allow each team to select one person to **rotate** and one person stay in place to **present** the game and record the critique. • Once the critiques have been completed, allow teams to reconvene and discuss the results of the critiques as well as their plans for modifications.
Reflection and Synthesis Prompts	• For the **rotator**: what aspects, if any, of the game(s) you critiqued would you like to also implement into your game? • For **presenter**: what was difficult about receiving the critiques. • For the **teams**: what aspects of the critiques sparked new ideas? What had you (potentially) thought about already (but decided against)? • What impact will the critique have on the design and creation of your prototype? • Do you think software developers go through this critique/modify process? If yes, why? If not, why not?
Formative Assessment Ideas	• Students can share, orally or in writing, their goals for each work day, what they accomplished, and next steps noting how the critiques impacted their choices.
Suggestions for Feedback and Support	• Use the Critique Protocol to help students prepare to give and receive feedback and their plans for addressing it.

	3. **IDEATE LEVEL 2**
Standards	CCSS.ELA-LITERACY.SL6.1.C; CCSS.ELA-LITERACY.SL6.1.D; CCSS.ELA-LITERACY.WHST.6-8.7; CCSS.ELA-LITERACY.RST.6-8.3; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.SL.6.4; CCSS.ELA-LITERACY.CCRA.W.7; CCSS.ELA-LITERACY.CCRA.W.9; CCSS.ELA-LITERACY.CCRA.SL.1; CCSS.ELA-LITERACY.CCRA.SL.2; MS-ETS1-1; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4
Students will be able to....	Identify concrete plans for implementing the second level of their game.
Ideas for activities	● Allow students to develop wireframes separately for Level 2, share their designs with their teammate, and collaboratively develop the final design. ● Allow students to generate a list of to-dos and NTKs for Level 2. The to-dos and NTKs should take their experience to date, observation of other teams' games, and feedback from their peers into consideration. The goal is to put students in the position where the development of Phase 2 is within reach and they have the basic tools and confidence needed to develop it.
Reflection and Synthesis Prompts	Allow teams to critique Level 2 plans and offer suggestions for improvement.
Formative Assessment Ideas	Allow teams to critique plans for future data analysis and offer suggestions for improvement.
Suggestions for Feedback and Support	Leverage the **Plus, Minus, Interesting** exercise to get a handle on how students absorbed the process so far. This exercise can provide an opportunity to correct some of the minus thinking.

MILESTONE #4: REPORT DEVELOPMENT

The goal of this milestone is to have students draft and finalize their presentations iteratively and improve them based on feedback from their peers (both internal and external to their classroom).



STEPPING STONES

DRAFT AND PRACTICE

Student teams draft their presentations and practice in front of other teams.

INCORPORATE FEEDBACK

Student teams improve their presentations based on feedback received by other teams (optional) and present to a different group of peers to gain more feedback.

RESOURCES

VIDEOS

[Schooled by Kids: Presentation Skills, Part 1]

ACTIVITIES

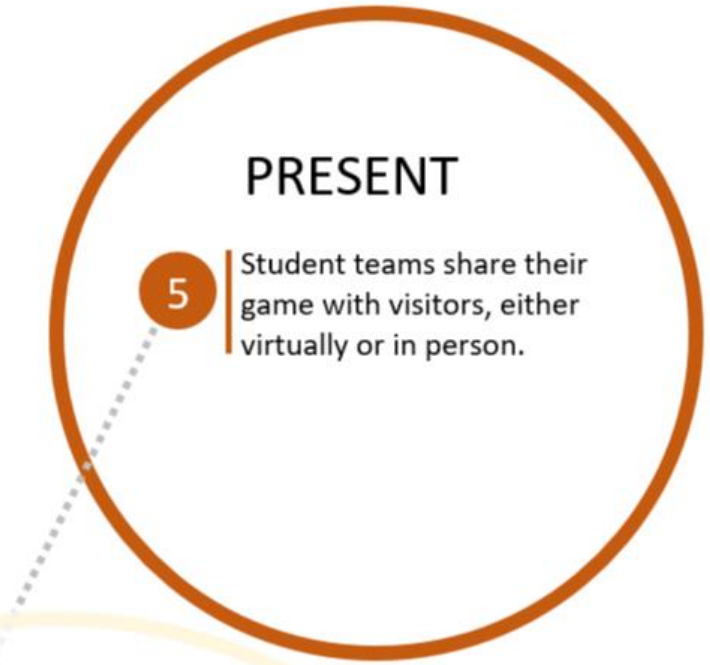
[Presentation Rubric for Evaluation]
[Presentation Overview]
[Feedback Assessment Form]
[3-2-1]

	1. **DRAFT AND PRACTICE**
Standards	CCSS.ELA-LITERACY.WHST.6-8.2; CCSS.ELA-LITERACY.WHST.6-8.5; CCSS.ELA-LITERACY.WHST.6-8.6; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.4; CCSS.ELA-LITERACY.SL.6.5; CCSS.ELA-LITERACY.CCRA.SL.4; CCSS.ELA-LITERACY.CCRA.SL.5; CCSS.ELA-LITERACY.CCRA.W.4; CCSS.ELA-LITERACY.CCRA.W.5; CCSS.ELA-LITERACY.CCRA.W.6
Students will be able to....	● Summarize the process they have taken to develop Level 1 of the game and their plan for Level 2. ● Present a draft presentation to other teams for initial review/critique.
Ideas for activities	● Work together to identify what the components of the presentation should be. Provide a Presentation Overview for the structure for the presentation and allow students to begin ideating. ● Have the students develop their presentations. ● Have a practice session where students present and receive feedback from other teams.
Reflection and Synthesis Prompts	● What did you enjoy about developing your presentation? ● How has watching other teams present improved your presentation? ● What presentation skills have you learned or refined as a result of this process? ● What presentation skills should we discuss in a mini-workshop?
Formative Assessment Ideas	Provide students with a Presentation Rubric that they will use to both critique other teams and self-assess.
Suggestions for Feedback and Support	● For students who need additional support in understanding how to give effective talks, share the following link: ● Schooled by Kids: Presentation Skills, Part 1

	2. INCORPORATE FEEDBACK
Standards	CCSS.ELA-LITERACY.SL6.1.C; CCSS.ELA-LITERACY.SL6.1.D; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.SL.6.4; CCSS.ELA-LITERACY.CCRA.W.9; CCSS.ELA-LITERACY.CCRA.SL.1; CCSS.ELA-LITERACY.CCRA.SL.2;
Students will be able to....	• Synthesize feedback from other teams and the instructor into the directives necessary to improve their presentation, to identify what feedback they will keep, what they will disregard, why they chose to do either, and (if they chose to accept the feedback) how they updated their presentations accordingly.
Ideas for activities	• Have students use the Feedback Assessment Form that includes columns for feedback received, whether they plan to keep or reject it, and (if kept) how they modified their presentation accordingly. • Optional: Have students who are not in the class (but roughly the same age) sit for a practice presentation and offer critiques. This should be done once students have received their first round of critiques from other teams and had a chance to make adjustments.
Reflection and Synthesis Prompts	• What was the best or most interesting piece of feedback your team received? • How did that feedback impact your project? • Was there any feedback you received earlier, that you did not incorporate, that you heard again in later critiques? If so, how did you address that?
Formative Assessment Ideas	• Leverage the 3-2-1 Exit Ticket to understand how well students can summarize their experiences developing/refining their presentations and what those summaries are. • Complete an edited draft of the presentation.
Suggestions for Feedback and Support	• It would be helpful for feedback to be recorded or written for students to keep with them while they work. The Feedback Assessment Form may be helpful in this way.

MILESTONE #5: FINAL PRESENTATION

The goal of this milestone is to have students present their work to an audience and receive feedback.



STEPPING STONES

PREPARE

Student teams make final preparations for their projects.

PRESENT

Student teams present their games to visitors either virtually or in person.

REFLECT

Students and teacher reflect on their original NTKs, their current understanding, and lessons learned.

RESOURCES

TOOLS

- Establishing Norms
- Standard Operating Procedure
- Presentation Rubric
- What, So What
- Plus, Minus, Interesting
- 3-2-1 Bridge
- I Used to Think – Now I Think

STEPPING STONES TO FINAL PRESENTATION

	1. **PREPARE**
Standards	CCSS.ELA-LITERACY.WHST.6-8.2; CCSS.ELA-LITERACY.WHST.6-8.5; CCSS.ELA-LITERACY.WHST.6-8.6; CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.SL.6.4; CCSS.ELA-LITERACY.SL.6.5; CCSS.ELA-LITERACY.CCRA.SL.1; CCSS.ELA-LITERACY.CCRA.SL.4; CCSS.ELA-LITERACY.CCRA.SL.5; CCSS.ELA-LITERACY.CCRA.W.3; CCSS.ELA-LITERACY.CCRA.W.4; CCSS.ELA-LITERACY.CCRA.W.5; CCSS.ELA-LITERACY.CCRA.W.6; CCSS.ELA-LITERACY.CCRA.W.8; CCSS.ELA-LITERACY.CCRA.W.9
Students will be able to....	Finalize their presentations and p**repare to share** their **design** approach, process, and **solution.**
Decision points	● When and where will students share their presentations? ● How will the presentation space be set up? ● What roles will individual groups/students play in the overall event for setup and takedown, as well as any other necessary tasks. ● **Note**: if you would like to engage students in the decision-making process, consider leveraging the Establishing Norms and Standard Operating Procedure frameworks.
Ideas for activities	● Film practice runs for students to self-assess. ● Have students self-assess based on the Presentation Rubric.
Logistics	● Create a guest list and send out invitations. ● Identify non-participant support to manage tech support, tend to guests, and/or troubleshoot when potential issues arise. ● **Decide how the audience will be invited, greeted, and seated/placed at the venue. Deciding whether to invite other guests, such as administrators and other classes, is also an important consideration.**

STEPPING STONES TO FINAL PRESENTATION

	2. PRESENT
Standards	CCSS.ELA-LITERACY.SL.6.1; CCSS.ELA-LITERACY.SL.6.2; CCSS.ELA-LITERACY.SL.6.4; CCSS.ELA-LITERACY.SL.6.5; CCSS.ELA-LITERACY.CCRA.SL.1; CCSS.ELA-LITERACY.CCRA.SL.4; CCSS.ELA-LITERACY.CCRA.SL.5;
Students will be able to....	● Present their games to an audience **to share what they have created.**
Decision points	● Depending on audience, some presentations may be virtual. Filming final presentations ahead of time would serve as a back-up for unanticipated absences and issues. ● How will students present: one group at a time or simultaneously while stakeholders move around? ● What will students do when they are not presenting? ● How can the experience be structured so that visitors can give meaningful feedback on the teams' approach/presentations.
Ideas for activities	● In-person or virtual presentation of prototypes or design ideas.
Logistics	● If you serve as the host of the event, it may be necessary to prepare a program. You also may need to step in when/if students cannot present for some reason. Enlisting the help of another adult will help to ensure the event runs smoothly. ● Outline the process for setup and take-down.

	3. **REFLECT**
Standards	CCSS.ELA-LITERACY.CCRA.SL.1; CCSS.ELA-LITERACY.CCRA.W.3; CCSS.ELA-LITERACY.RI.6.1; CCSS.ELA-LITERACY.SL.6.1.A; CCSS.ELA-LITERACY.SL.6.1.B; CCSS.ELA-LITERACY.SL.6.1.C; CCSS.ELA-LITERACY.SL.6.1.D; CCSS.ELA-LITERACY.WHST.6-8.2.D; CCSS.MATH.CONTENT.6.SP.A.2;
Students will be able to....	**Reflect with the teacher on the project and their original NTKs through collaborative discussions and writing.** A key focus of this activity will be to help them understand how much they have advanced since Milestone #1.
Ideas for Reflection on Work Product and Process	● Allow students to develop **individual written reflection** and then confer with their team to round out their recollection of key activities. Once everyone has submitted his or her reflection, **follow the written activity up with a whole-group discussion on the project.** ● Leverage the following the **Plus, Minus, Interesting** worksheet to support these discussions/writings: ● Do the **What, So What?** exercise to frame the discussion of how the knowledge and experience gained as a result of the project will be leveraged in the future and explicitly state the next steps. ● Deliberately and strategically encourage students to continue with the project by suggesting that they work with a second data set/questions in their own time as an independent study.
Reflection on Presentation	Discuss how the presentations evolved over time and how the event with visitors went, overall.
Formative Assessment Ideas	Allow teams to critique plans for future data analysis and offer suggestions for improvement.
Ideas for Reflection on NTKs and Key Knowledge	● **Individually revisit NTKs and write about what they have learned, what they have yet to discover, and a plan for addressing the latter.** ● Leverage the following frameworks to support reflection: ○ **3-2-1 Bridge** ○ **I Used to Think – Now I Think** ● Provide reflective prompts for discussion including: ○ **I was surprised when** I learned/understood that**...** ● **I'm still wondering about...**

Works Cited

Hammond, Z. (2015). *Culturally Responsive Teaching and the Brain: Promoting Authentic Engagement and Rigor Among Culturally and Linguistically Diverse Students.* Thousand Oaks, CA: Corwin.

Yamaguchi, R., Madrigal, V., Eaton, C., Hall, A., & Burge, J. (2021a). *Equitable learning experiences for girls of colour in STEM+CS: Nurturing independent learner behaviours through a CS learning ecosystem.* Manuscript submitted for publication.

Yamaguchi, R., Madrigal, V., Eaton, C., & Hall, A. (2021b). *Supporting Black and other girls of colour in computer science through structural, instructional and curricular supports.* Manuscript submitted for publication.

Buck Institute for Education. (2017). *Out of the Gate*. http://ootg.pblworks.org/ootg/