K–12 Pair Programming Toolkit

advancing health equity etr.

Our Team

We are a team of educators, curriculum developers and researchers. We have expertise in education, psychology, anthropology, and computer science. We have been teaching and conducting research on pair programming with middle school students for over 10 years—investigating what pair programming really looks like at this age, exploring the benefits of students programming in pairs, and studying gender and cultural differences in pair programming and how it relates to learning. The classes in our studies have been electives during school and voluntary afterschool programs where students program interactive games, first learning the software and then designing and making a game.

Acknowledgments

This Toolkit was made possible by the National Science Foundation grant #1420596 awarded to ETR. Principal Investigator Jill Denner and Co-Investigator Shannon Campe would like to acknowledge the educators and research assistants who worked directly with students and provided valuable insight into effective activities and strategies: Luci Basor, Sara Shimel, Justen Time, Yethzéll Díaz and Julie Adams. We also would like to acknowledge those who worked on various phases of research during the project: Emily Green, David Torres, Valerie Muñoz, Huimee Sanchez, Imelda Melchor, Jacque Alonso, Jason Ou and our advisors: Florence Sullivan, Brigid Baron, Omar Ruvalcaba, Patrice Gans and Susan Rivas. Lastly, this would not be possible without Co-Investigator and university computer science faculty Linda Werner, who introduced us to pair programming long ago.

Suggested Citation

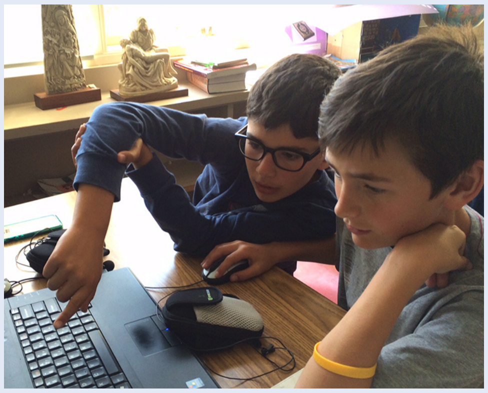
Campe, S., Green, E. & Denner, J. (2019). K-12 Pair Programming Toolkit. ETR, Scotts Valley, CA.

This material is based on work supported by the National Science Foundaation under grant #1420596.

Contents

Introduction

In the world of technology professions, females, Blacks and Latinx continue to be substantially underrepresented. Through our work with partners and our own research we have designed and supported efforts to increase diversity in computer science education. One of the strategies we've frequently used and studied is *pair programming*. Out of this work grew the desire to take what we have learned and share it with educators as a collaborative computing strategy to engage and address a greater range of students and their varied styles of interacting.


Pair programming is a collaborative learning strategy to engage and address a greater range of students and their varied styles of interacting.

Educators may want to pair students doing programming because they believe it will make the process more fun, less intimidating and encourage collaborative learning. This intuitively makes sense. But educators don't always know the best way to pair students or the conditions under which pair programming has benefits for learning. We have studied what pair programming looks like on a deeper level, how it may or may not increase students' engagement and learning, and which students gain benefits from pair programming.

What is Pair Programming?

Pair Programming (PP) is a pedagogy where two people work side by side with one computer, monitor, keyboard and mouse—each trading roles as Driver and Navigator. The Driver controls programming actions with the mouse and keyboard and the Navigator provides input to programming actions, looks for mistakes and accesses resources. This collaborative strategy is known to be effective both in K-12[1, 2] and at the college level.[3] There are resources for higher education faculty,[4] but tools are limited for K-12 educators.

Educators may choose to use PP in their classroom for a variety of reasons ranging from a desire to show the "social side" of computing to not having enough computers for all students. PP can be used for a range of tasks, including exploring new software, debugging, closed-ended tasks with explicit directions, and open-ended design projects. Tasks that include explicit directions will have a more defined role for the Navigator, who can guide the Driver through steps. More open-ended tasks may require more support from the teacher to help pairs work together successfully. Effective pairs give and receive input, exchange Driver and Navigator roles to build in each other's expertise and help each other think through problems.[5] Partners will need to learn how to co-construct and implement design and programming ideas since the Driver is the only one touching the mouse and keyboard. PP can be used sporadically for specific tasks, or as often as every day. If you are using PP for a short-term project, you may not use the same methods pairing students as if you are expecting them to work together for a long period of time. Pairs working together for a longer duration will take more planning and support to be effective as they build their relationship. But the strategies to teach and support them remain the same.

Research Basis

The recommendations in this Toolkit are based on research conducted by our team and others. That research includes descriptions of what PP looks like in middle school, as well as studies that highlight the characteristics of partners that may influence how they interact and what they learn. For example, interactions that involve both support and challenges, and that cause students to reflect on and explain their thinking, are most likely to benefit learning.[6]

The nature of collaboration can vary greatly. Some partners talk a lot, while others rely more on nonverbal communication. Some focus primarily on the task, and others talk about their friends, family, movies, or schoolwork. This social talk has been found to have benefits for learning.[7] Partners often have disagreements, and while some argue about what they want to do or how to fix something in their program, others allow the driver to make the decision.

When there are arguments, some resolve these disagreements and land on a respectful mutual solution. Others fail to find compromise within the conflict and their partnership deteriorates. Cultural differences can explain some of the variation in levels of disagreement and antagonistic actions.[8] Throughout this Toolkit we highlight further research on the various dimensions of PP.

Purpose of Toolkit

This Toolkit is a How-To Guide from conception to implementation for educators who are interested in using PP in their classes. Successful PP requires careful planning and preparation to build strong partnerships, as well as support and scaffolding to persist when there are challenges. However, the nature and extent of the preparation and support will vary depending on the goals and duration of specific PP projects.

This Toolkit is based on our research and programs with middle school students, but we have used the strategy with younger and older K-12 students. Out of this work grew the desire to take what we have learned and share it with educators as a collaborative computing strategy to engage and address various student learning styles. We identify issues to consider when making decisions about whether or how to use PP in your classroom and provide suggestions and resources for implementing it. You are welcome to utilize any or all parts of the toolkit in your computing classes.

Remember that PP is just another way of working collaboratively on a project. As students do their pair work, be sure to utilize your own favorite strategies to guide students through the process. For example, check in with them regularly about how they are doing. Assist them with resolving conflicts as necessary. Acknowledge positive examples as you see pairs work together. And encourage them to have fun!

Why Use Pair Programming

The first question to ask yourself is, why have your students pair program? It is important to be clear about your motivation for using PP, since that will determine the amount of time and focus that you will spend on a variety of PP activities. We have identified four main reasons: (1) to teach the core practice of collaboration in computer science, (2) to increase students' interest in computing, (3) to teach students programming skills, and (4) to increase student participation when there are software or hardware limitations. Below we explore these four reasons in more detail.

Pair programming can promote collaboration, increase computing interest and skills as well as address issues of limited computer access.

1 Teach Collaboration

Collaboration is an essential skill for all students. It's increasingly important in computing as the Computer Science for All (CS4All) movement takes hold and more teachers are expected to teach computer science. Collaborative computing is described in the K–12 CS Framework as "the process of performing a computational task by working in pairs and on teams," which "requires individuals to navigate and incorporate diverse perspectives, conflicting ideas, disparate skills, and distinct personalities."[9]

Similarly, the Computer Science Teachers Association (CSTA) standards recommend that students learn how to seek and incorporate feedback from team members and users to refine a solution that meets user needs.[10] Using PP provides students an opportunity to learn collaboration skills by practicing Driver and Navigator roles, giving and receiving feedback and discussing diverse perspectives.

Additionally, PP can encourage independent problem solving. Having a partner means students are more likely to seek help from a peer before reaching out to the teacher. While implementing PP, the teacher can establish class norms for problem solving that include first asking a partner and then other peers/pairs in the class.

2 Increase Interest in Computing

There are still demographic groups that are underrepresented in computing, such as female, African American, and Latinx students.[11] The reasons for this are complex, but some contributing factors are stereotypes and pedagogies that reinforce beliefs that computing is a solo endeavor.

With PP, teachers can challenge that belief and show how computing involves creativity and curiosity as partners bounce design and programming ideas off each other. Studies have shown that while PP can increase the retention of all students in college computing classes, it is particularly beneficial for female students.[3]

3 Teach Programming Skills

Students working in pairs demonstrate greater increases in programming knowledge and computational thinking compared to their peers who programmed alone.[1] Students also are more likely to say they enjoy programming and to persist in problem solving while programming with a partner compared to working alone.[2] In addition, PP has been found to result in greater increases in computer programming knowledge in middle school girls compared to boys in the same studies.[12, 13]

4 Address System Limitations

Some schools and community programs do not have enough computers and/or software licenses for every student to work on a computer at the same time. Teachers can address this challenge by using PP. Two students are able to work together using only one computer.

Pair Programming Computer Station Configuration

Monitor must be visible to both students.

Support materials must be accessible.

There must be enough room for both students to sit in front of the monitor.

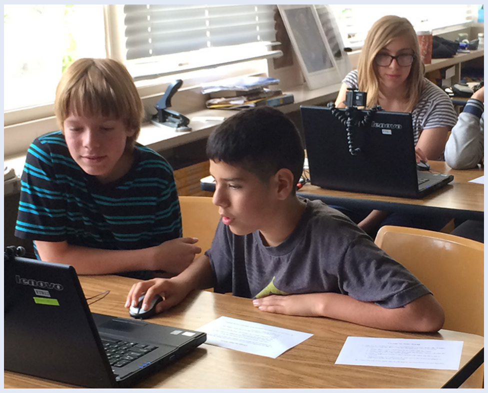
How to Pair Students

You now have an idea of why you want to implement PP, so the next step is to address how you will pair your students. Pairing students is possibly the biggest consideration when doing PP in the classroom. While some pairs will work easily together, others will need scaffolding to be successful.[14]

You may want to take different approaches to pairing students depending on your goals for the PP project. When you assign pairings, you can use data on students' attitudes, prior knowledge, and feelings of closeness to their partner, as well as class observations of pair interactions. One way to collect this information about students is to administer a paper or online survey (see Pair Programming Activities section for examples).

If you want to prioritize friendships, we recommend having students pair program for a short time with 3-4 partners (e.g., using self-selection or a random strategy such as counting off numbers to pair up), then asking them to privately write down their three top partner preferences. You can then review their preferences in light of your own observations to confirm pairings.

The following section lists various possible goals and suggests ways to pair students to help them achieve these outcomes. Your identified goals will help guide your pairing decisions and scaffolding needs for implementing PP in your classroom.


Pairing students is possibly the biggest consideration when doing Pair Programming in the classroom.

Research Says...

Partners reporting stronger partnership at the onset of pairing can make for more compatible pairs throughout working together.[13]

To Build Collaboration Skills

Learning to collaborate requires a willingness on the part of both students to interact. If your goal is for students to build collaboration skills, it's helpful to pair students who have similar attitudes toward collaboration and a perception that they have an equal or higher level of programming skill than their partner (not one partner thinking they are less skilled).

Prior relationship may be less important when the goal is to promote collaboration. While friends may want to work together, if they hold very different views about collaboration and have different levels of prior experience, this can undermine their ability to interact productively.

Research Says...

Pairs who initially have a more positive attitude toward collaboration spend more time talking about the programming project, and less time with a third person present.[15] However, students who have a more positive attitude toward collaboration relative to their partner are more likely to experience decreases in programming knowledge.[1]

To Encourage Interest in Computing

If your goal is to encourage students to be more interested in computing, then pairing students with a similar attitude toward working with a partner appears to be more important than pairing them based on a similar skill level. It's okay for the partners to have discrepancies in prior computing knowledge, but the benefits they accrue will be different. For example, when a more experienced student is paired with a less experienced one, the one with more experience may gain more confidence and a positive attitude toward computing, while the one with less experience will gain more programming knowledge.[1]

Research Says...

Students report greater compatibility with their partner when they see them as having equal or higher skill level, a similar work ethic, and different learning styles. Actual skill level and self-esteem did not influence compatibility.[16] Students report producing their best work when paired with someone who had a similar level of programming confidence.[17]

To Build Programming Skills

If your goal is to help students learn how to program, pairing them with friends may be the best strategy. This is because feeling close to one's partner can lead to the kinds of interactions that increase computer programming knowledge regardless of whether or not they have similar levels of prior programming knowledge before working together.

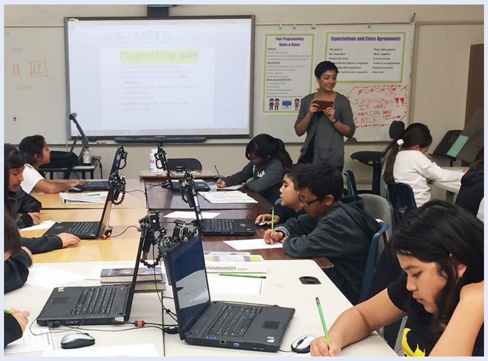
Research Says...

Students who felt closer to their partner before they started PP had greater increases in computer programming knowledge.[12]

While students with less experience may gain more knowledge than their more experienced partner, it is unlikely there will be negative effects for either one. In fact, having similar levels of programming confidence may be more important than similar levels of knowledge.

Teaching & Support

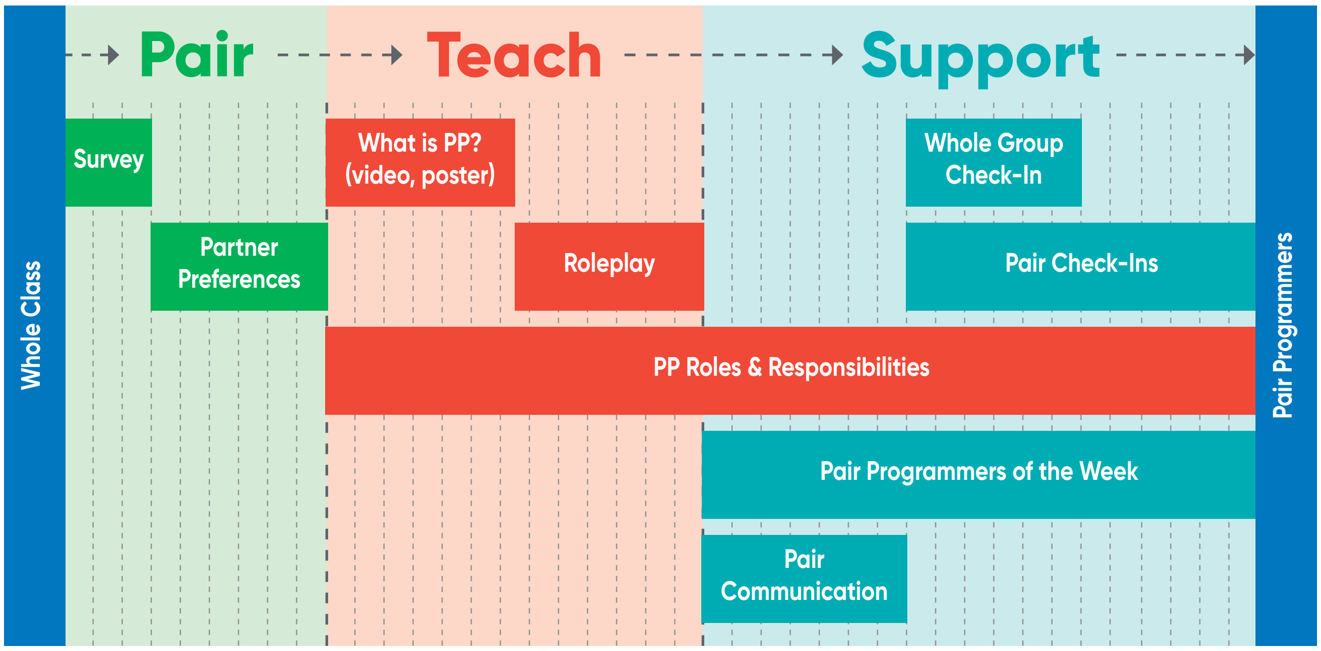
Once you have decided to use PP with your students and have thought about how to pair them, the next steps are to teach them the PP basics and provide guidance and support throughout the collaborative programming process. This section includes an Activity Map for implementing PP in the classroom, an overview of the basic principles of PP, the characteristics of effective PP behaviors, and strategies for how best to support PP students.


To become effective pair programmers, students need practice and your continued support.

Steps to Implement Pair Programming

1) **Pair** students
2) **Teach** PP basics, emphasizing partners' roles and responsibilities
3) **Support** pairs with team building, acknowledgments, check-ins and conflict resolution

Activity Map



This Activity Map displays the three broad areas involved with implementing PP in the classroom: pairing students, teaching the basics of PP and the partners' roles and responsibilities, and providing support to pairs.

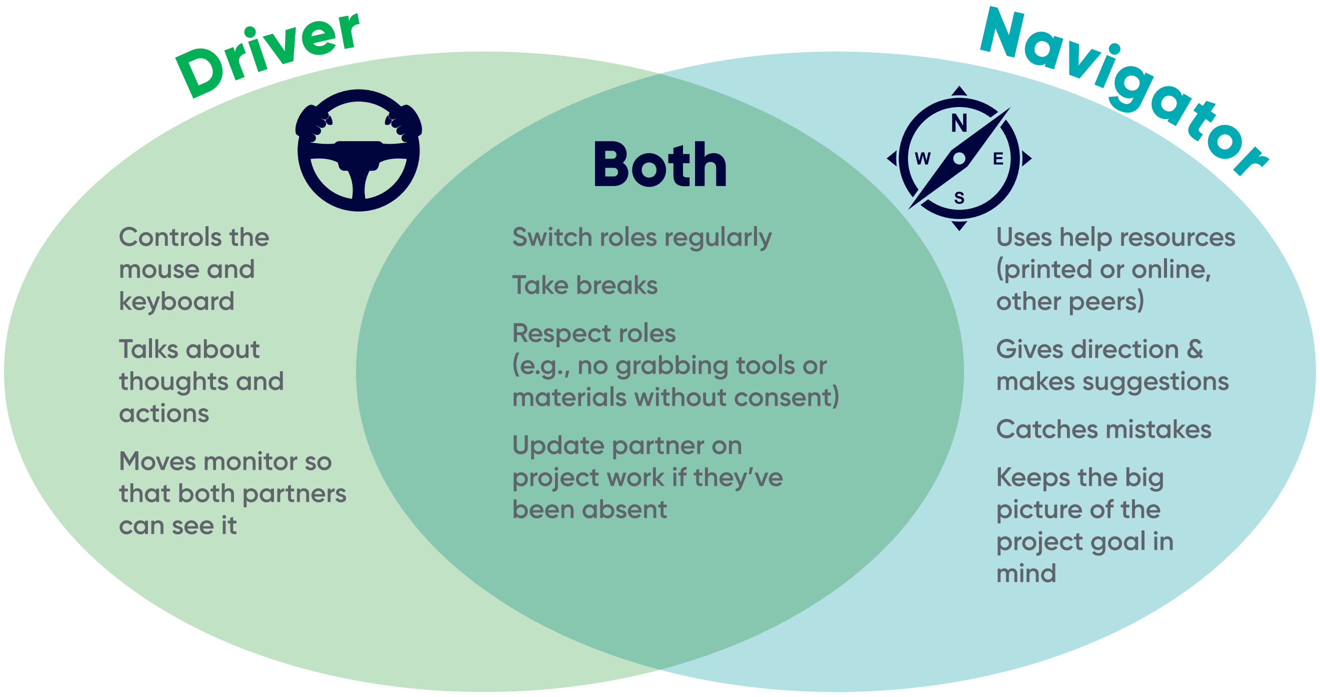
Sample lesson plans and materials can be found in the Pair Programming Activities section. You'll want to choose activities according to the goals you have for your students as well as how they fit into your overall curriculum time and plans.

Principles of Pair Programming

Lesson plans and materials provide more ideas for class activities, but here we outline the principles to emphasize as you teach and support PP. You'll find a **Principles of Pair Programming Quick Reference** guide in the Appendix.

Roles & Responsibilities. The Driver controls the mouse and keyboard; the Navigator monitors the programming and offers suggestions. The partners have additional responsibilities that support the process: they should be actively cooperating, talking about ideas and solutions, switching roles in a way that builds upon each others' expertise and listening and responding to each other in a respectful way.

Partners should be actively working together: talking about ideas and solutions, switching roles in a way that builds upon each others' expertise, and listening and responding to each other in a respectful way.


Driver
Navigator
Both
N
W
E
S
Controls the mouse and keyboard
Talks about thoughts and actions
Moves monitor so that both partners can see it
Switch roles regularly
Take breaks
Respect roles (e.g., no grabbing tools or materials without consent)
Update partner on project work if they've been absent
Uses help resources (printed or online, other peers)
Gives direction & makes suggestions
Catches mistakes
Keeps the big picture of the project goal in mind

Teamwork Skills to Support PP. In addition to the individual roles and responsibilities required for successful PP, you'll want to pull from your own classroom norms and collaborative learning knowledge to enhance your students' class-generated **Pair Programming Roles & Responsibilities** poster. Guide your students to include some of the following ideas to support teamwork:

- Listen
- Compromise
- Share ideas
- Give feedback
- Check for agreement
- Respectfully disagree
- Check in about roles, engagement and programming decisions
- Step-back sometimes if you tend to take over
- Step-in by speaking up if you tend to stay quiet

Value of PP. Help students see the value of programming with a partner by emphasizing some of the following benefits as you pair, teach and support PP:

+ Built-in help—students help each other; don't need to wait for teacher
+ Fun—it can be more fun to work together instead of alone
+ 2 heads are better than 1—generate more design and problem-solving ideas; catch mistakes
+ Communication skills—practice a new "coding" language while talking with your partner
+ Real-life skills—teamwork is used in school and work to create a better final product

Strategies to Support Pair Programming

As you work with students, keep in mind that PP may look different than anticipated and partnership dynamics will vary. Interactions can appear serious, playful, talkative, or quiet as students think independently or take a short break so they can re-engage more effectively.

Observe how partnerships evolve over time. This will help you see what aspects of PP need support, recognize any disengagement and conflict early enough to intervene effectively, reinforce partners' positive behaviors and assess their progress on the project.

Research Says…

½ Half of pairs' time together was spent interacting about their game project, with limited variation due to gender, school or other factors.[15]

⅓ A third of their time together, students didn't appear to be interacting at all.[15]

This section provides key reminders of what to emphasize with your students and suggests strategies for helping your students with programming and pair challenges.

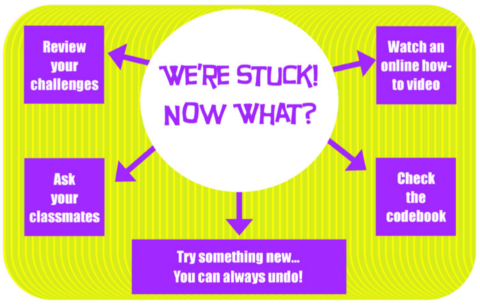
Key Reminders. In addition to implementing the recommended activities, support your students by grounding classwork in the class-generated **PP Roles & Responsibilities**. Emphasize effective PP behaviors throughout programming sessions and through informal or formal check-ins with pairs using the following reminders as a guide:

+ Remind pairs that there is no failing at PP—they're practicing a new skill.
+ Encourage them to have fun creating and programming together.
+ Affirm students for what they are doing well; include specific examples when possible.
+ Use words other than "good" to describe positive PP behaviors (e.g., "effective," "beneficial," "positive").
+ Emphasize teamwork and communication. Remind them to say "we" and "us" not "I" and "me."
+ Ask students to briefly share out a couple of reasons why it's good to program with a partner.
+ Encourage more assertive students to "step-back" in the pair—to listen and respond to their partner. Encourage less assertive students to "step-in"—to talk about their ideas and help solve programming challenges.
+ Remind pairs to switch Navigator and Driver roles to build upon each others' expertise and monitor for imbalances in time or power in the Driver role.
+ Challenge them to work together effectively. Ask students to come up with ideas on how to improve their partnership.

Helping with Programming Challenges. Help strengthen partnerships and promote independent problem solving by encouraging students to rely on each other (in and outside their own partnership) when they encounter a programming challenge.

You can do this by teaching them specific steps to take before soliciting your help. For example:

+ What's the challenge?
+ Check the codebook.
+ Watch an online how-to-video.
+ Ask classmates for help.
+ Try something new.



A poster like this (in Appendix) can be introduced to students at the beginning of PP and referred to consistently throughout their project.

Switching Roles. There is debate about whether partners should routinely switch Navigator and Driver roles. Some say that it is necessary for ensuring equity.[18] We observed that regulating the role switches can sometimes undermine deep thinking and collaboration. Our analysis of role switching that was not strictly regulated showed that effective pairs exchange their time as Driver and Navigator to build on each other's expertise.[5] Another study with a similar "semi-free" switching structure found that although partners switched roles less as time went on, they also negotiated more about the switches they did do (i.e., interacted more).[19]

One strategy is to regulate role switching at the beginning and then let the pairs decide how often they switch. For example, you can start out by having them switch every 15-20 minutes so that each person gets to try out each role. When there are more time-defined, closed-ended tasks, another strategy to consider is to have them switch roles based on the task.

If you want to actively encourage role switches, you can use verbal reminders or a timer. When using a timer, make sure it's visible to the whole class and have students stand up and physically switch seats as they take on their new role. Enforcing consistent role switching can be challenging and time consuming but it will ensure that *some* role switching will happen for pairs that do not (or cannot) make it happen on their own.

If pairs resist switching roles, you can introduce more structure than you started with. For example, you can start using a timer if you hadn't before. Or you can continue with your same approach but observe the pairs more closely to ensure programming is equitable. Before regulating the timing of the role switching, ask the students if they have a good reason for staying in one role more often. Just because one partner has the keyboard more often, there's not necessarily an imbalance. There is often more engagement than is apparent from across the room. Even if a pair appears to be quiet, the Driver may be implementing something the partners have discussed.

Whether you choose to simply encourage or strictly enforce the switching of roles, be intentional and transparent about what is expected and how the switching will take place. Check in regularly with pairs to increase engagement and equity, and to reduce problematic conflict. Once you establish a consistent policy, your level of monitoring will be more manageable.

Troubleshooting Disengagement and Conflict. Conflicts within pairs *will* arise and there are strategies to help mitigate this. Often, what looks like conflict from across a classroom is actually just an intense exchange in which partners are challenging each other in ways that lead to successful problem solving. However, some conflicts will arise that partners need help to resolve. There are several strategies you can use.

Pay special attention to either partner's repeated disengagement as a form of silent conflict within the pair. A typical example is when the Navigator is disengaged (e.g., looking around, not paying attention to the screen or partner, talking to others or looking upset). Some examples and questions you might ask them are:

Research Says...

7% Seven percent of the the time pairs engaged in disruptive behaviors that held up programming.[15]

- The Navigator appears to be bored and just waiting for their turn to be Driver. Did the pair communicate their roles and how they are dividing tasks?
- The Navigator appears irritated. Is the Driver including the Navigator in decision making, or is the Driver ignoring or shutting down the Navigator's input?
- The partners are not communicating with one another. Are both partners talking about what they are doing, helping direct their partner and/or asking questions?
- The Navigator does not appear to be tracking what is going on. Do they understand that paying attention to the "big picture" is one of their responsibilities?
- One partner is getting much more time as the Driver than the other. Do they have a reason for this, or do they need more reminders to switch roles?

Once you have more information on what is happening with each partner and the pair, work with students to implement applicable elements of the Principles of Pair Programming (roles, responsibilities, teamwork) to support their partnership.

Deciding to Re-Pair Students. There are instances when you may need to reorganize pairs due to a partner's excessive absences or to conflict the pair cannot resolve to stay on task. You will need to assess which students would work well together and which original pairs would be least hindered when making changes. Focus on what students can bring to the new pair and ground your decision in the Principles of Pair Programming.

We are excited to see more and more teachers working to engage and include all students in computing. Pair programming can be one way to do that by teaching and guiding students in an approach to collaborative learning that offers support as they take on new programming challenges. We hope that the research and lessons learned that are described in this Toolkit will help teachers to optimize the potential of pair programming in their classroom(s).

References

1 Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education,* 46(3), 277-296.

2 Liebenberg, J., Mentz, E., & Breed, B. (2012). Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT). *Computer Science Education,* 22(3), 219-236.

3 Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC),* 4(1), 4.

4 National Center for Women & Information Technology (2009). *Pair Programming-in-a-Box: The Power of Collaborative Learning.* Retrieved from **https://www.ncwit.org/resources/pair-programming-box-power-collaborative-learning**.

5 Denner, J., Green, E., & Campe, S. (under review). Learning to program in middle school: How pair programming helps and hinders intrepid exploration.

6 Kuhn, D. (2015). Thinking together and alone. *Educational Researcher,* 44 (1), 46-53.

7 Leman, P. J., Skipper, Y., Watling, D., & Rutland, A. (2016). Conceptual change in science is facilitated through peer collaboration for boys but not for girls. *Child development,* 87(1), 176-183.

8 Ruvalcaba, O., Werner, L., & Denner, J. (2016). Observations of Pair Programming: Variations in Collaboration Across Demographic Groups. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 90-95). ACM.

9 K-12 CS Education Framework (2016). Retrieved from **https://k12cs.org/**.

10 Computer Science Teachers Association (2017). CSTA K-12 Computer Science Standards, Revised 2017. Retrieved from **http://www.csteachers.org/standards**.

11 National Center for Education Statistics (2016). Digest of education statistics 2014, 50th edition. Available at: **https://nces.ed.gov/pubs2016/2016006.pdf**.

12 Hartl, A. C., DeLay, D., Laursen, B., Denner, J., Werner, L., Campe, S., & Ortiz, E. (2015). Dyadic instruction for middle school students: liking promotes learning. *Learning and individual differences,* 44, 33-39.

13 Zhong, B., Wang, Q., & Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior,* 64, 423-431.

14 Tsan, J., Rodríguez, F. J., Boyer, K. E., & Lynch, C. (2018). I Think We Should...: Analyzing Elementary Students' Collaborative Processes for Giving and Taking Suggestions. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 622-627). ACM.

15 Campe, S., Denner, J., Green, E., & Torres, D. (in press). *Pair Programming in Middle School: Variations in Interactions and Behaviors. Computer Science Education.*

16 Williams, L., Layman, L., Osborne, J., & Katira, N. (2006). Examining the compatibility of student pair programmers. In *Agile Conference, 2006* (pp. 10-pp). IEEE.

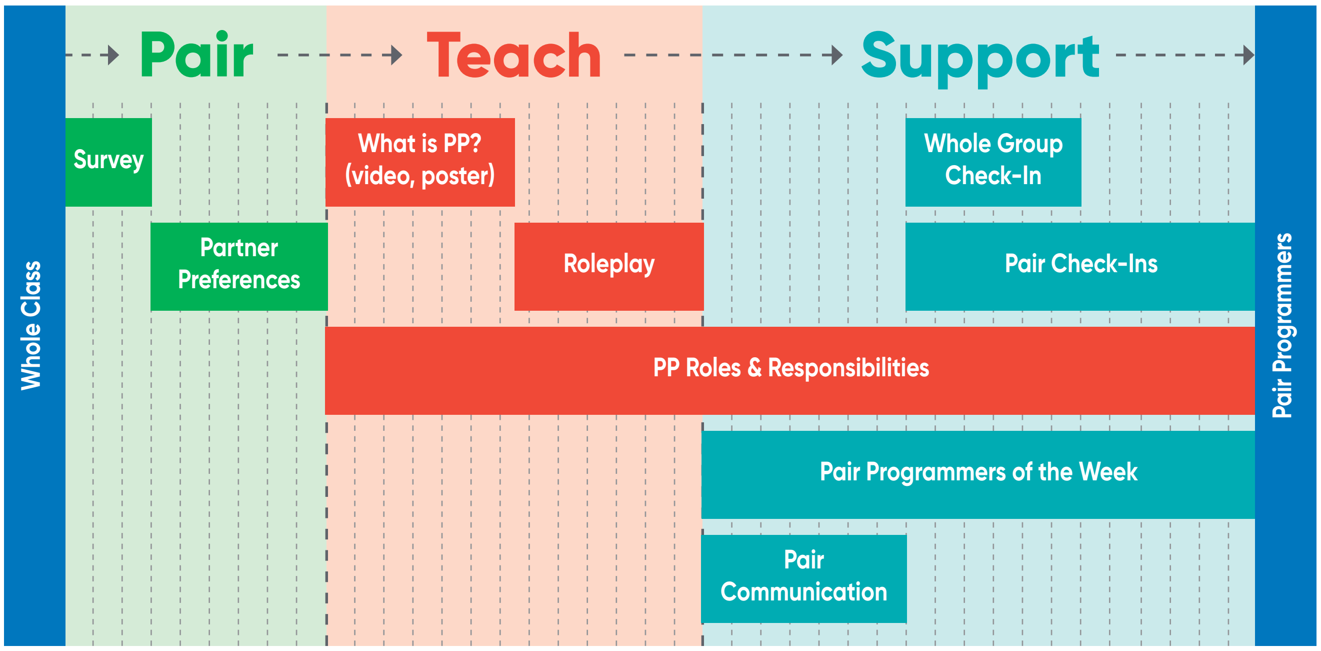
17 Thomas, L., Ratcliffe, M., & Robertson, A. (2003). Code warriors and code-a-phobes: a study in attitude and pair programming. In *ACM SIGCSE Bulletin* (Vol. 35, No. 1, pp. 363-367). ACM.

18 Lewis, C. M., & Shah, N. (2015). How equity and inequity can emerge in pair programming. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 41-50). ACM.

19 Zhong, B., Wang, Q., Chen, J., & Li, Y. (2017). Investigating the period of switching roles in pair programming in a primary school. *Journal of Educational Technology & Society,* 20(3), 220-233.

Pair Programming Activities

Activity Map



This material is based on work supported by the National Science Foundation under grant #1420596.

Surveys

5–20 minutes

Purpose

- To collect information on students' computing experience, skills and/or knowledge, confidence with computers, attitudes toward collaboration and level of friendship with their partner that will assist you in pairing and supporting students.

Materials

- Survey for each student (online or paper)

Preparation

- Create survey (on or offline) with questions and response options
- If online: Secure access for each student to take survey on a computer
- If on paper: Print survey for each student

Activity Steps

1. Have students individually take survey that includes groups of questions selected from categories outlined below.
2. Review survey responses and partner preferences (optional) to create pairings.

Categories of Survey Questions and Responses

You'll find sample surveys on the following pages. These are examples of surveys to help you pair students by collecting information in these general categories:

1. Computing Experience
2. Programming Knowledge
3. Confidence with Computers
4. Attitude toward Collaboration
5. Level of Friendship with Partner

These are samples only. You may want to generate questions and statements more suitable for your class situation.

1 Survey on Computing Experience

To capture data about how your students use computers and other technologies, create relevant questions based on what is currently used at your school and in their homes. Below are some sample questions and response options to use and/or modify for your students.[1]

How often do you use technology (phone, computer, tablet) to do the following things?	**Never**	**Rarely**	**Sometimes**	**A lot**
Text/email friends and/or family				
Use a social networking site (Instagram, Facebook, etc.)				
Play computer/video games				
Search for information on the internet				
Use tutorials/drill and practice software				
Write reports, essays, etc.				
Design or program a computer/video game				
Learn and use a new programming (coding) language				
Create presentations				
Create spreadsheets/databases				
Create/work on your own online journal or blog				
Take material you find online—like songs, texts or images—and remix it into your own creation				
Create your own videos				
Create or work on your own webpage				
Edit photos or graphics in a computer program				

1 Adapted from Martin, C.K., Barron, B., Matthews, J., & Stringer, D. (2014). In B. Barron, K. Gomez, N. Pinkard, & C.K. Martin (Eds.). *The Digital Youth Network: Cultivating New Media Citizenship in Urban Communities*, pp. 203-235. Boston, MA: MIT Press. ISBN: 978-0262027038

2 Survey on Programming Knowledge

To discover information about students' previous programming skills and knowledge, our projects used examples specific to the free Alice interactive 3-D programming environment that captured elements of computational thinking (e.g., abstraction, algorithmic thinking).[2] You can recreate a similar survey in your own classroom based on whichever programming tool you use. Take screen shots from the software you are using and craft 4-6 questions and response options like the following example.[3]

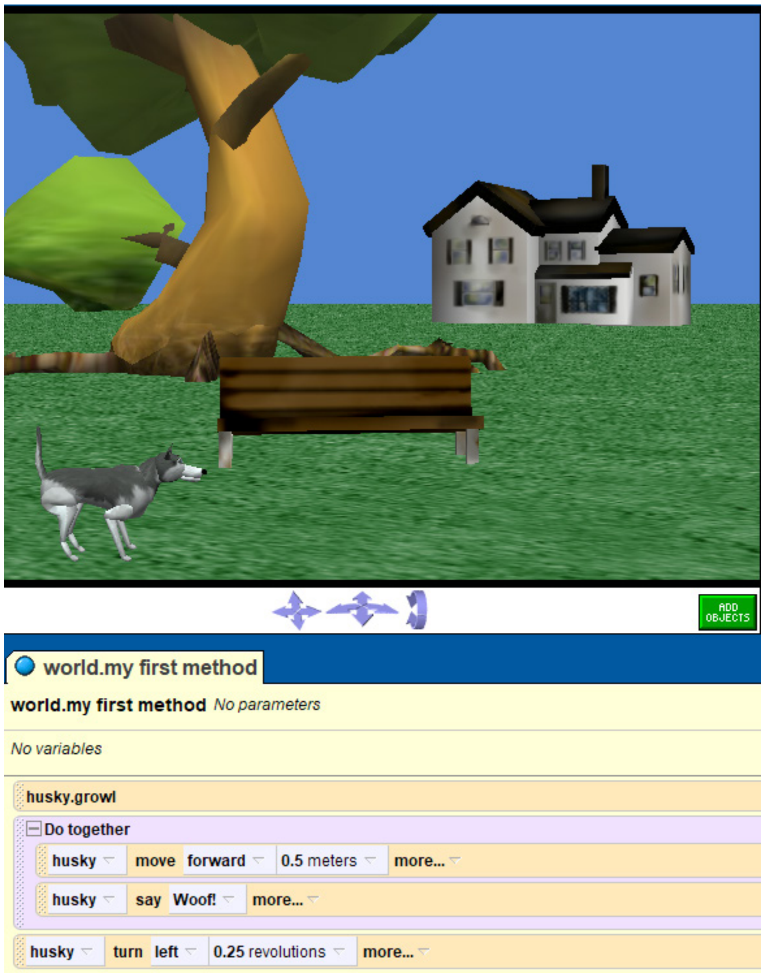
Question: If you were to play this method in Alice, which of the following best describes what would happen?

a. Husky would growl, move forward, say "Woof!" and turn left all at the same time.

b. First husky would growl, then husky would move forward and say "Woof!" at the same time, and finally husky would do a ¼ turn to the left.

c. First husky would growl, then husky would move forward, then husky would say "Woof!" and finally husky would do a ¼ turn to the left.

d. First husky would growl, then husky would move forward, say "Woof!" and do a ¼ turn to the left at the same time.



```
world.my first method
world.my first method  No parameters
No variables
husky.growl
Do together
    husky  move  forward  0.5 meters  more...
    husky  say  Woof!  more...
husky  turn  left  0.25 revolutions  more...
```

2 www.alice.org

3 Werner, L., Denner, J., Campe, S., & Kawamoto, D.C. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM: New York, NY USA, 215-220. DOI=http://dx.doi.org.oca.ucsc.edu/10.1145/2157136.2157200

3 Survey on Confidence with Computers

To collect information on students' confidence working with computers, see the example questions and response options below. Note that these sample statements include both positively and negatively worded statements.[4]

How much do you agree with each statement?	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I am good with computers.					
I feel confident about my ability to use computers.					
I am NOT the kind of person who works well with computers.					
When I use technology, I think about how it works.					
I like the challenge of learning how to use a new technology.					
I don't want to learn any more about computers than I need to know to make them work.					

4 Survey on Attitude Toward Collaboration

As noted in the Toolkit, students' attitudes toward collaboration can play a big role in how they interact as partners. To give you more information on what attitude they bring into the partnership to help you pair and support students, see the example questions and response options below. Note that these sample statements include both positively and negatively worded statements.[5]

How much do you agree with each statement?	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I learn more when I work with others.					
I have more fun when I work with others.					
When more than one person works on a project, we have better ideas and make better things.					
Working with other people on projects is mostly a waste of time.					
I often end up doing all the work when I am on a team.					
I prefer to work alone.					

[4] Adapted from Barron, B., Walter, S.E., Martin, C.K., & Schatz, C. (2010). Predictors of creative computing participation and profiles of experience in two Silicon Valley middle schools. Computers & Education, 54, 178-189.

[5] Adapted from Martin, C.K., Barron, B., Matthews, J., & Stringer, D. (2014). In B. Barron, K. Gomez, N. Pinkard, & C.K. Martin (Eds.). *The Digital Youth Network: Cultivating New Media Citizenship in Urban Communities,* pp. 203-235. Boston, MA: MIT Press. ISBN: 978-0262027038

5 Survey on Level of Friendship with Partner

This survey requires that the student have an identified partner.[6] A teacher can use this information on potential pairings *prior* to final pairings or *after* final pairings to better understand partners' relationship to one another, which can be helpful for supporting their work together.

How much do you agree with each statement?	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I feel happy when I am with my partner.					
If my partner had to move away I would miss him or her.					
When I do a good job at something my partner is happy for me.					
Sometimes my partner does things for me or makes me feel special.					
I can count on my partner for help.					
I think about my partner even when my partner is not around.					
My partner is my friend.					
My partner and I spend time together outside of this class.					

6 Adapted from Bukowski, W. M., Hoza, B., & Boivin, M. (1994). Measuring friendship quality during pre-and early adolescence: The development and psychometric properties of the Friendship Qualities Scale. *Journal of Social and Personal Relationships,* 11(3), 471-484.

Partner Preferences

5 minutes

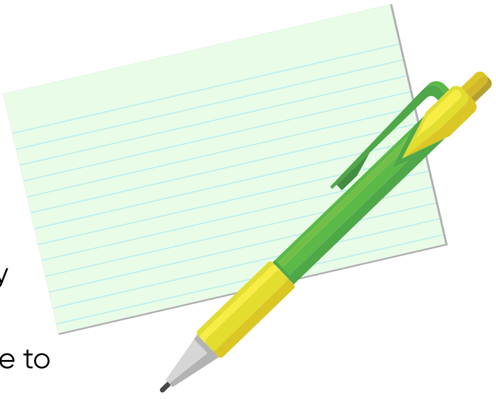
Purpose

- To collect information on students' preferences for partners, giving students a stronger voice and creating buy-in to their partnership, and to assist in pairing and supporting students.

Materials

- Pencils/pens (1 per student)
- Index cards or scratch paper (1 per student)

Activity Steps



1. Tell students that it is time for them to write down their preferences for a programming partner.
2. Remind them to think about both the people they have worked with previously as well as classmates they think they would work well with. Remind them to think about who they would work well with, which may not necessarily be who they are good friends with.
3. Have students spread out so they are not sitting too close to anyone else.
4. Hand out pencils/pens and index cards/scratch paper and ask each student to write:
   a. Their name at the top
   b. Number 1, 2 and 3 down the left side
   c. Their #1, #2 and #3 choices for a partner
5. Tell them NOT to show their choices to anyone else.
6. Tell them you will do your best to pair them with their preferences and you will announce the pairs at the next programming session.
7. When students are done, collect cards/paper and make sure their own names are on them along with their 3 partner preferences.

Teacher Note

After class look at students' preferences. Compare these with your own notes on who worked well together and consider any survey data and considerations mentioned in the Toolkit.

What is Pair Programming?

30 minutes

Purpose

- To introduce what pair programming is and how it is used.
- To explain the benefits of pair programming.
- To have students generate a poster listing the roles and behaviors for effective pair programming.

Materials

- Computers with internet access
- Teacher computer, projector/screen
- White board, screen or chart paper
- Markers
- Pair Programming Video
- **Principles of Pair Programming Quick Reference** (in Appendix)

Preparation

- Select and view the student video you choose (see options below; download prior if necessary)
- Watch teacher-focused video/s (optional)
- Review **Principles of Pair Programming Quick Reference** to have a good understanding of roles and behaviors

Activity Steps

▶ Introduction

1. Tell students that they are now going to learn about pair programming, which is just like it sounds—programming on the computer with a partner. Today they will come up with their own class list of what makes a partnership or a team work well.
2. Ask students what they know about pair programming. Briefly explain that the Driver operates the mouse and keyboard while the Navigator helps direct the Driver, watches for mistakes and keeps track of any materials they are using. Explain that beyond the basic roles of Driver and Navigator, each pair can have its own way to work together as long as both partners are engaged, working as a team to complete their programming goals and having fun.
3. Explain that they are going to watch a video to learn how they will practice pair programming in class with a partner.

▶ Video Questions

1. Write up the following questions for all to consider:
   - What are the main jobs for the Driver and the Navigator?
   - What should BOTH partners DO?
   - What should BOTH partners NOT do?
   - Why is pair programming worth doing?

Teacher Note

If you think it helps your class to have them write down what they see as they watch the video, have them do so.

▶ Play Video

▶ Debrief & Poster Creation

1. After the video, post headings for Driver, Navigator and Both on the white board, screen or chart paper.
2. Using **Principles of Pair Programming Quick Reference** (at the end of this lesson) to prompt student responses, have students share individually or via pair share, what they saw in the video based on the questions above. Remind them to think about what has worked or not in other team projects in or out of school and on or off the computer.
3. Record responses under each heading *(Driver, Navigator, Both)* in students' own words as much as possible.
4. Explain to students that their comments will be used to generate a **Pair Programming Roles & Responsibilities** poster. The poster will be a reminder of what makes pair programming successful so they can practice the skills throughout the class.
5. Help students narrow down the list of roles and behaviors, including grouping similar items. The list doesn't have to include everything but should have the basics around roles and respectful, positive, engaged teamwork.
6. Summarize activity by reminding students that each time they start on the computer they will need to decide who starts in which role first and remember to switch so that each of them gets practice in each role. Emphasize that the most important thing is that both partners are respectful and involved and that they communicate with each other!

Teacher Note

Create **Pair Programming Roles & Responsibilities** poster out of what class generated. Be sure to include main roles of Driver and Navigator and what both partners should be doing.

Video options:

For students:

Title: Pair Programming
Author: code.org
About: Middle school girl pair programmers; do's and don'ts of PP; why do PP
Time: 2:50 minutes
URL: https://www.youtube.com/watch?v=q7d_JtyCq1A

Title: Introduction to Pair Programming, version 2
Author: North Carolina State University
About: College pair programmers; good description of the strategy; benefits of pair programming; long video so consider showing segments
Time: 10:02 minutes
URL: https://www.youtube.com/watch?v=rG_U12uqRhE

For teacher:

Title: Teaching CS Fundamentals: Pair Programming
Author: code.org
About: Teacher's role; reasons to pair program
Time: 1:47 minutes
URL: https://www.youtube.com/watch?v=sxToW3ixrwo

Title: Tips for Pair Programming
Author: CSTeachingTips.org
About: Tips for teachers, including encouraging equitable roles
Time: 4:16 minutes
URL: https://www.youtube.com/watch?v=TWj78n4ZuMY&t

Roleplay

20 minutes

Purpose

+ To review and reinforce students' roles and responsibilities for effective pair programming.

Materials

+ Teacher computer and mouse
+ **Pair Programming Roles & Responsibilities** poster
+ **Principles of Pair Programming Quick Reference**
+ Screen or board/markers
+ Pencils
+ Scratch paper
+ Programming instructions for a selected task
+ Timer

Preparation

+ Review **Principles of Pair Programming Quick Reference** for effective pair programming behaviors.
+ Recruit and prepare a student to be your programming partner for the roleplay. Talk to this partner about the following regarding the roleplay:
  - He/she will be the Driver and you will be the Navigator.
  - Pretend that the Driver was absent the previous class.
  - Do *[insert relevant programming task]* as programming partners.
  - Be sure you both talk loud enough so all students can hear.
  - Be sure you both exhibit good and bad PP behaviors.

Activity Steps

▶ Introduction

1. Acknowledge that students have been working in pairs for *[insert days, classes, etc.]* now. Note examples of what you have seen that are "good" pair programming behaviors.
2. Present students with the question: *"Why do we use pair programming when there are plenty of computers for everyone?"* Make sure the following are included in their answers:
   - Having a partner gives you a built-in helper to solve problems you encounter.

- Many people find computers more fun and less lonely/isolating if they get to work with someone else.
- It is easier to learn all the new words and codes in computer programming if you have someone to talk to about them.
- Working with a partner helps you develop good teamwork skills, which are beneficial for school, future jobs, careers and life in general.

3. Explain to students that they are going to get a chance to observe a skit/roleplay of you and a student pair programming. After the roleplay they will give feedback on examples of what they saw you doing that was good/effective pair programming and what was not.
4. Hand out a piece of scratch paper and pencil to each student and tell them that they will be expected to report out on their observations after the roleplay ends.

▶ Perform Roleplay

1. Position the computer and binder so it's difficult for you, the Navigator, to see the screen and sit with your roleplay programming partner. Be sure the other students can clearly see and hear the roleplay.
2. Start the roleplay by talking to your partner about how they were absent last time, so you want to update them about what you did in their absence. Tell them you finished *[insert task]* and you can show your partner the completed assignment after you complete this one. Offer for them to be the Driver this time since you did all the programming prior.
3. Start directions for *[insert task]*.
4. Be an active Navigator who is having fun and getting along with their partner, doing many positive PP behaviors with a few not so good behaviors such as being disrespectful about something the Driver says/does, trying to grab the mouse, etc. (Refer to **Principles of Pair Programming Quick Reference** prior for ideas.) Focus on positive as well as negative behaviors.
5. After about 5 minutes of working and exhibiting roles and behaviors (have fun with it!), stop the roleplay and thank your Driver for helping you demo for the class.

▶ Debrief

1. Ask the class to tell you what they observed during the roleplay and record their responses for all to see.
2. Discuss what they observed and make links to the **Pair Programming Roles & Responsibilities** poster they created. Reiterate and reinforce positive roles and behaviors you have observed in class.
3. Emphasize that they need to continue to do their best at pair programming, including regularly switching roles with their partner so they get to perform each role as equally as possible.

Pair Communication Activity #1: Draw What I Say

20 minutes

Purpose

+ For students to develop collaborative skills and use descriptive language relevant to working together on the computer.

Materials

+ **Pair Programming Roles & Responsibilities** poster
+ **Principles of Pair Programming Quick Reference** (for your reference)
+ Copies of the model drawings (1 of each model per pair)
+ Blank paper
+ Pencils
+ Timer
+ Debrief questions posted for class

Preparation

+ Configure space so that pairs have enough room to sit facing each other, with a binder or book propped up in between and flat space to draw figures.

Activity Steps

▶ Introduction

1. Tell students that they are going to do an activity *off the computer* with their partner called "Draw What I Say." The goal is for the team to generate a complete, accurate copy of a drawing described by one partner (the Describer) to the other (the Drawer). The Drawer will not be able to see the original drawing during the activity.

Teacher Note

Keep instructions brief to get students going on the activity and assist as needed.

2. Ask students why they think they are doing this activity. Summarize that they will be practicing teamwork and building communication skills with their partner. Reinforce the idea that teamwork is very important to their goal of co-creating a project on the computer.

▶ Demonstration

1. Review and demonstrate the following roles with another adult or student using a binder as a barrier between the two partners.

Describer selects a figure and describes it to partner with words only. The Describer cannot show anything to the Drawer or touch their paper or pencil but can watch to see how to make corrections if needed.

Drawer tries to draw an exact copy of the figure that the Describer has hidden behind a barrier by listening to instructions and asking clarifying questions.

2. Tell (and show) students that when the Describer is finished giving instructions, and the Drawer feels finished with the drawing, the partners should take down the barrier and compare the two figures. Then they can switch roles and try describing and drawing another figure.
3. Remind them that it's okay to laugh about the drawing not matching exactly for the demonstration! The exercise isn't about being a good artist, it's about practicing ways to listen and communicate with your partner in different roles.
4. Check for understanding by asking if there are any questions before starting the activity. If a few students are confused, get the rest of the group going and help those students individually.
5. Tell students they can start as soon as the Describer picks up Figure #1, blank paper, pencil and binder. If they finish before the timer goes off, they can switch roles and try a new figure.

▶ Facilitate Activity

1. Have pairs get started. When all pairs are settled, set timer for 5 minutes.
2. Rotate around the room and help as needed. Prompt pairs to communicate as necessary to complete at least one drawing. Encourage them to do more than one drawing.
3. When timer goes off, have pairs switch roles and do another round of the activity with a new figure. Let new Describer choose whether to go one step higher in difficulty.
4. With 5 minutes left of activity time, have the students come to a stopping point with their drawings and bring them together as a group.

▶ Debrief

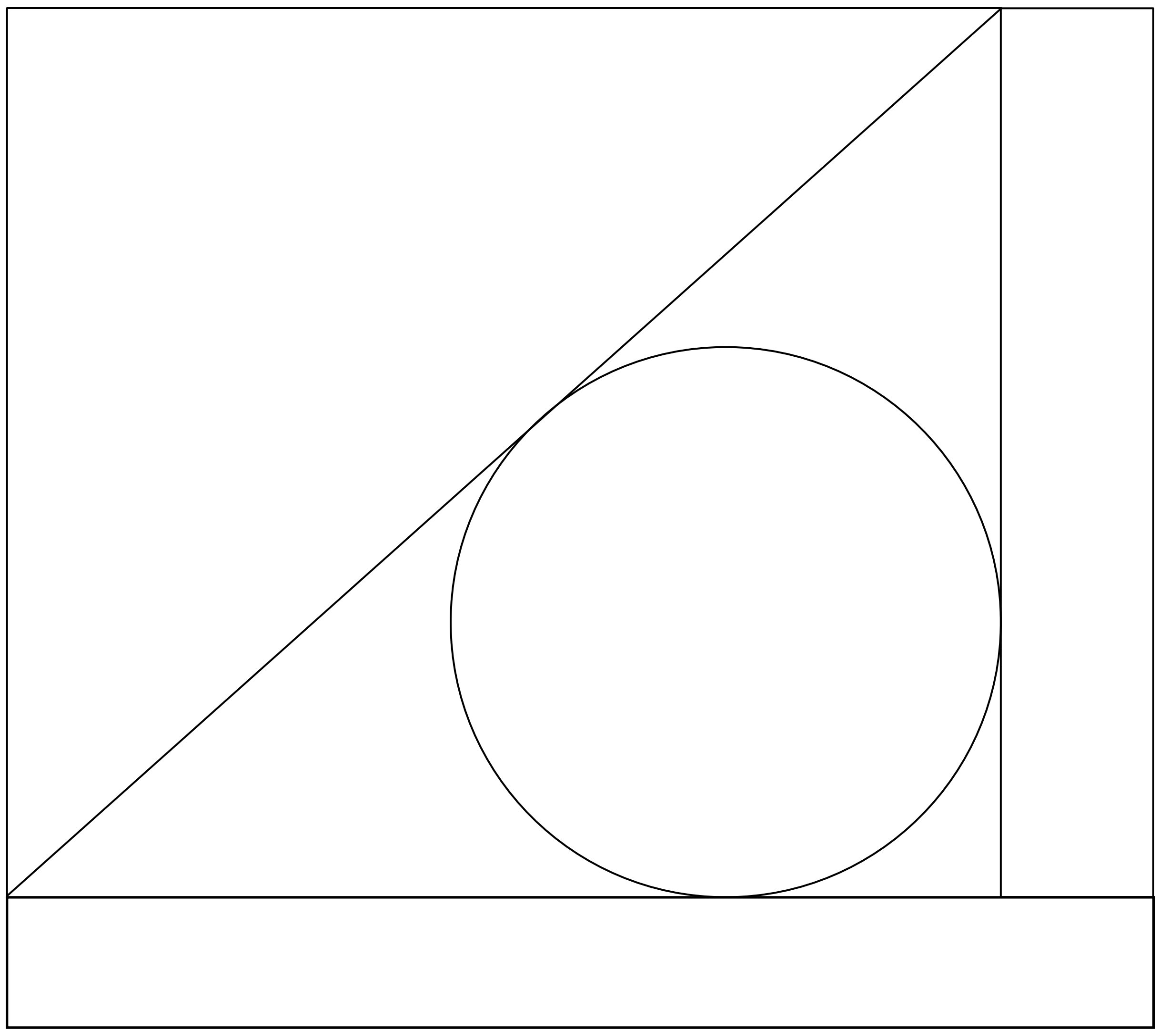
1. Read (or have a student read) the following debrief questions to the class and have pairs report out to the whole group:
   - What PP roles did these roles remind you of? (Navigator=Describer; Driver=Drawer)
   - What did you do that helped you make exact copies of the figures?
   - What kinds of words or instructions did partners come up with to draw the right shapes in the right place?

- How does this relate to what you are doing while pair programming? (e.g., patience with partner, asking questions, using descriptive language, giving clear directions)

2. Summarize the activity by explaining that it is important for students to communicate using a language for precisely describing things while programming so that partners can understand one another and succeed at pair programming. Remind them of good teamwork skills like being patient with one another, respecting each other's efforts while you are learning new skills and roles. Refer to **Pair Programming Roles & Responsibilities** poster as applicable.

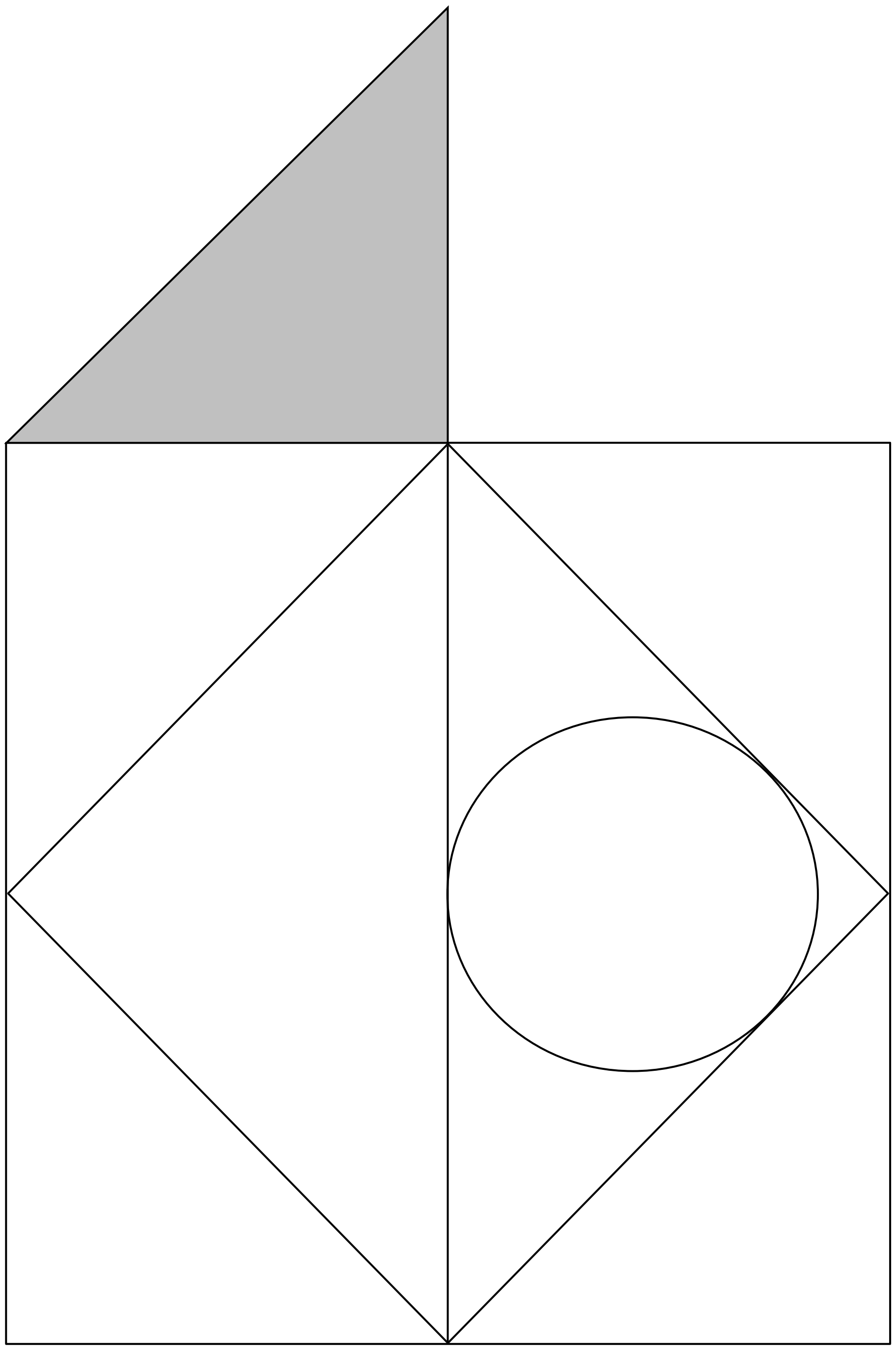
Draw What I Say Figures

Figure #1 (Level: Easy)



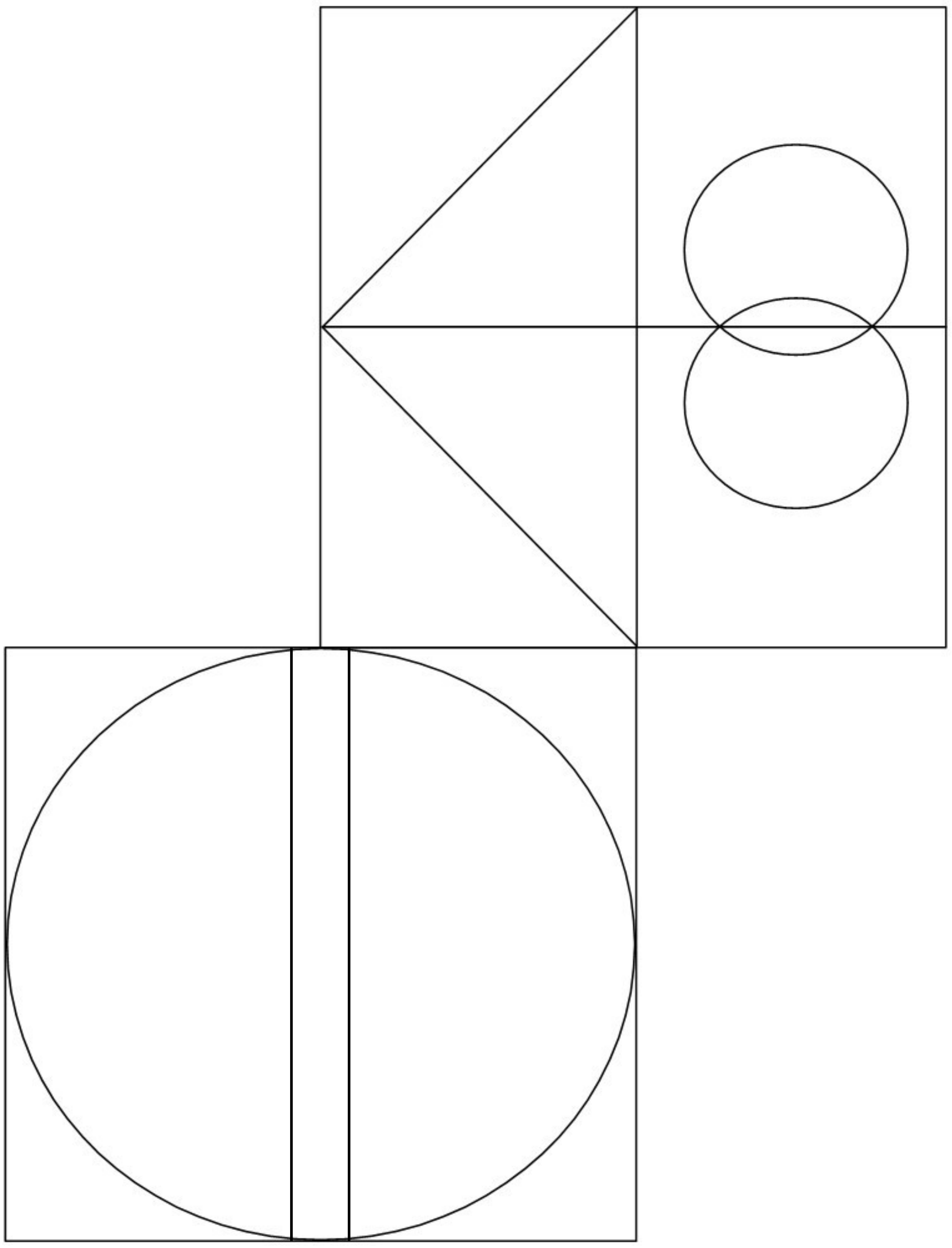
Draw What I Say Figures

Figure #2 (Level: Medium)



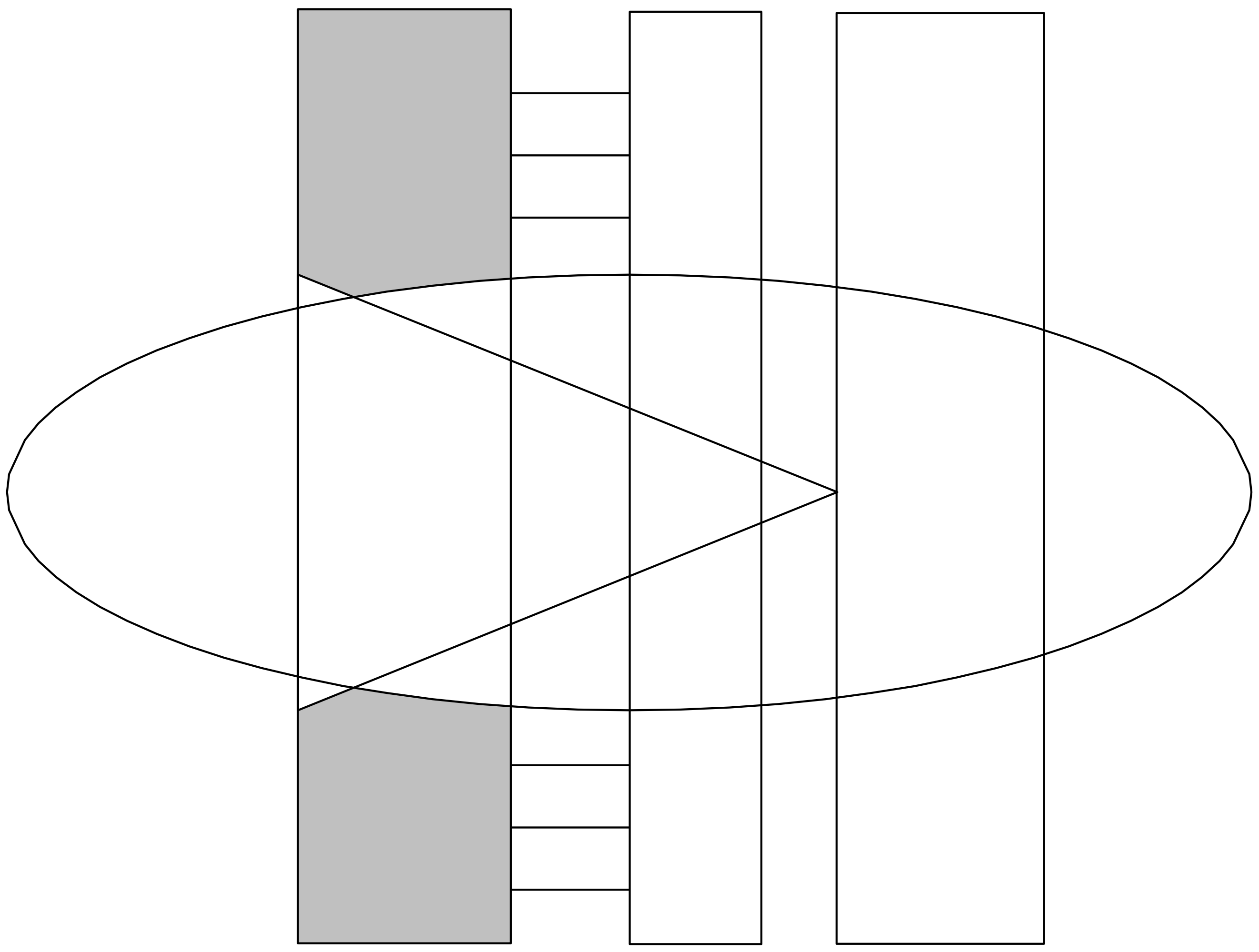
Draw What I Say Figures

Figure #3 (Level: Difficult)



Draw What I Say Figures

Figure #4 (Level: Very Difficult)



Pair Communication Activity #2: Telephone Architects

25 minutes

Purpose

+ For students to develop collaborative skills and use descriptive language relevant to working together on the computer.

Materials

+ **Pair Programming Roles & Responsibilities** poster
+ **Principles of Pair Programming Quick Reference**
+ Student binders or books
+ Activity roles and rules posted (optional)
+ Kids' 3-D building toys (e.g., Legos, Tinker Toys, K'nex) divided into bundles of two identical sets of pieces for each pair of students (e.g., for 20 students you'll need 10 bundles of two identical sets)
+ Timer

Preparation

+ Configure space so that pairs have enough room to sit facing each other, with a binder or book propped up in between and flat space to build structures.

Activity Steps

▶ Introduction

1. Tell students that they are going to do an off-the-computer activity called "Telephone Architects" where they get to problem-solve and practice teamwork to build communication skills with their partner. Reinforce the idea that teamwork is very important as pair programmers.
2. Tell students that the goal is to create a copy of a 3-D structure with their partner.

Teacher Note

If students are not familiar with the building toys, show them a couple different ways in which they connect. Keep instructions brief to get them going on the activity and assist as needed.

3. Explain roles and instructions. One person is the Architect and the other is the Builder.

Architect's role: To design and build your own design of a 3-D (not just lying on the ground flat) structure using the pieces in their bag.

- Hide your 3-D structure from your partner.
- Use ALL the pieces.
- Build your structure in 5 minutes.
- Describe to your partner how to build an EXACT copy of your 3-D structure.

Builder's role: To recreate an exact copy of the 3-D structure that your partner designed and built in their "secret design site."

- Use a bundle of identical building pieces.
- Listen carefully to your partner's instructions.
- Ask your partner clarifying questions.
- Do not look at the structure your partner built.
- Build your structure in 10 minutes.

4. Ask students how these "Telephone Architect" roles relate to PP roles. (Navigator=Architect; Driver=Builder). Help as needed to make connections with the roles.
5. Explain the following rules:
   - The Architect CANNOT point to or touch any of the Builder's pieces. They can ONLY describe in words which pieces to use and how to put them together.
   - The Architect CANNOT change the structure in the "secret design area" once the Builder begins working.
   - The Architect CAN look at what the Builder is making to help describe and correct their actions.
   - When both partners agree that they think the Builder is finished with the structure, take down the binder screen and compare the master design with the copy.

6. Tell them that as time permits, they will switch Architect/Builder roles and try another new structure.
7. Check for understanding and remind students:
   - Get as far as you can, it doesn't matter if you finish.
   - It's a team project—both roles are very important.
   - The Architect can begin making a structure as soon as they pick up their pieces.

▶ Facilitate Activity

1. Have pairs decide who the Architect is and get a bundle of 2 sets of matching building toy pieces to bring to their "building site."
2. When all students have their pieces, set timer for 4 minutes. When it goes off tell them they have 1 more minute for Architects to finish their structures. Remind them that they need to use ALL the pieces.

3. When all Architects have started describing their structures to Builders, set the timer for 10 minutes.
4. Rotate around room to monitor and prompt pairs to communicate as necessary to achieve success in duplicating the structure.
5. When timer goes off, have students finish and clean up and move on to debriefing the activity.

Teacher Note

If you have 10-15 minutes in addition to saving time for debrief activity, have students do another round of designing and building before moving on to debrief.

▶ Debrief

1. Post the question: What did you and your partner do to be successful at copying your structure?
2. Ask partners to talk for 2 minutes about it. Tell them that you will ask them to share their answer with the class.
3. After 2 minutes, ask as many pairs as time allows to report out their answers to the question. Link to PP behaviors as much as possible. Prompt them to share any words or instruction they used to communicate better. Refer to **Pair Programming Roles & Responsibility** poster as applicable.
4. Summarize activity by reminding students that it is important to communicate in a way their partner understands so that they can be successful pair programmers. Remind them that it takes good teamwork skills, like being patient with one another and respecting each other's efforts while each of you learn new things.

Pair Programmers of the Week

5 minutes

Purpose

- To acknowledge programming partners to help reinforce positive pair programming behaviors for all students.

Materials

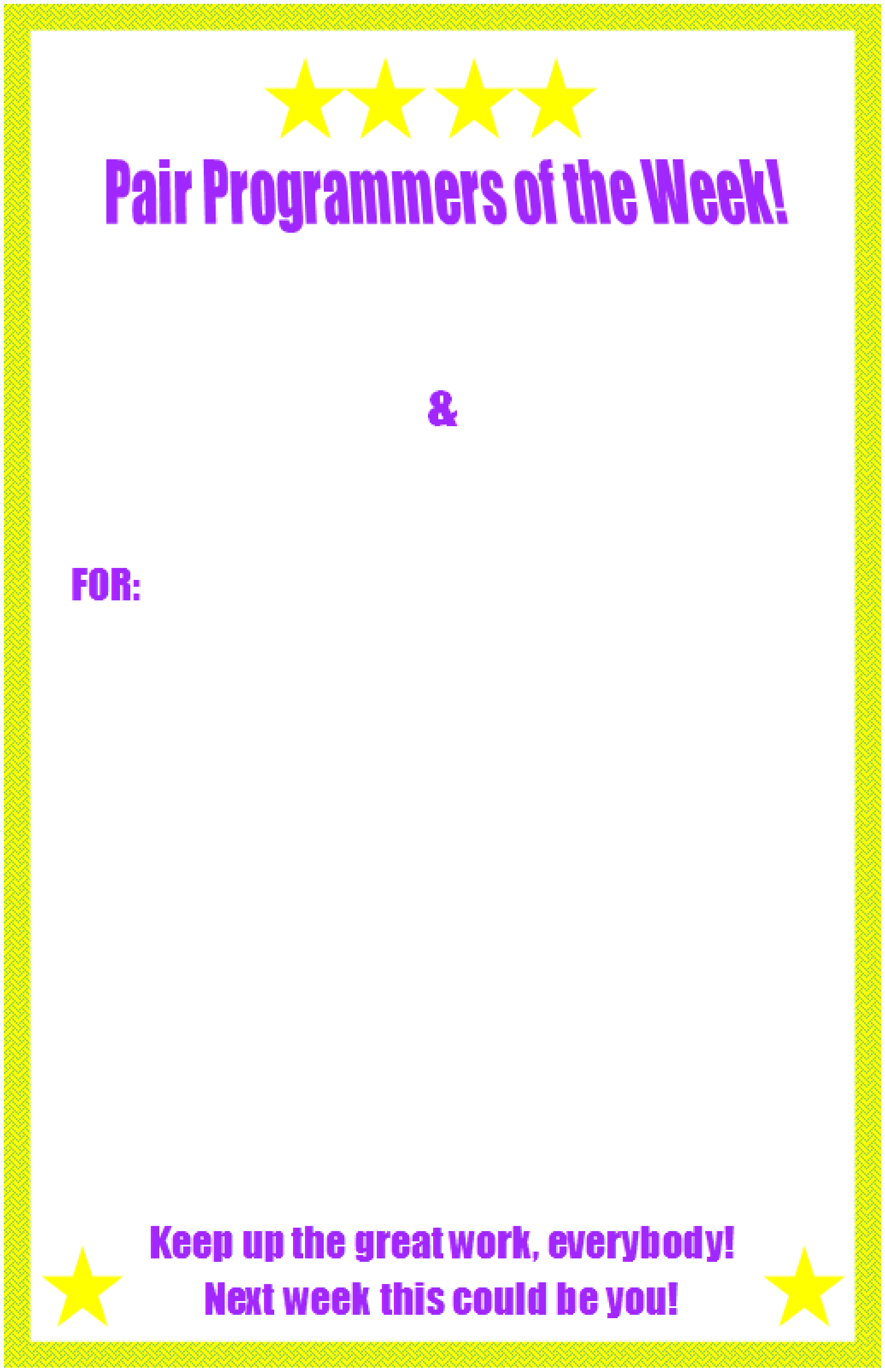
- **Pair Programmers of the Week** identified from previous session
- Designated space in class to post names and examples (e.g., connected to **Pair Programming Roles & Responsibilities** poster, additional poster or wall space)
- Markers, paper (depending on how you will post pair names and examples each week or session)
- Prizes (optional)

Preparation

- Observe students working together each programming session, including taking notes if necessary, to identify a pair to acknowledge the following session.
- Identify 2–3 specific examples of positive behaviors (e.g., Anna (Driver) respectfully asks Aliya (Navigator) for help to remember how to code something; how Aliya is always involved and watching what is going on as Navigator).
- Try to acknowledge all pairs throughout the course of their time programming together.

Activity Steps

1. In front of whole class, verbally acknowledge **Pair Programmers of the Week**—noting examples of why they are PP of the Week. Link examples with **Pair Programming Roles & Responsibilities** poster as applicable.
2. Post names and examples.
3. Distribute pair prizes (optional).



Pair Programmers of the Week!

&

FOR:

Keep up the great work, everybody!
Next week this could be you!

Whole Group Check-In

15 minutes

Purpose

- For students to reflect on their PP behaviors and reinforce the effective behaviors listed by the class.

Materials

- **Pair Programming Roles & Responsibilities** poster
- Chart paper or white board
- Markers of different colors (1 per pair)

Preparation

- Prior to class, create a grid on chart paper or board to list effective PP behaviors and rate performance. See attached example.

Activity Steps

▶ Introduction

1. Explain to students that it's now time to check in about how they are doing with PP these past few weeks/classes.
2. Show them the chart you have posted and explain:
    a. On the left are all the programming behaviors from the poster they created. Across the top is a rating for how well they are doing each of those behaviors: Needs Improvement, Doing Okay, Doing Great!
    b. Each student will get a marker. It doesn't matter what color it is.
    c. One partner from each pair will get up first to put a dot on the chart rating how well they think they are working with their partner.
    d. After the first group is done, students will hand off the marker to their partner, who will then make their own mark.
    e. We will then look at the chart as a class and discuss what most people feel they are doing well and what they need to improve.
3. Check for understanding and answer any questions.

▶ Facilitate Activity

1. Have the first group (one partner from each pair) get up and mark dots.
2. Have the second group (other partner from each pair) get up and mark dots.

▶ Debrief

1. After all students have rated themselves, lead a discussion based on where the dots are clustered. The focus should be on using this data to become a stronger, more effective team. Ask students what they see in the chart/dots and synopsize where the strengths and weaknesses are.
2. Depending on what the students report, some possible debrief questions/discussion scenarios include:
   - **Most dots are in "Doing Okay" or "Doing Great!" for a particular category.** Focus on how they are achieving that.

     What are specific examples of what you do or say for a particular role or responsibility?

     What are you thinking when you are tempted to [do an undesirable behavior]? How do you stop and do something different?

     Why did those of you who rated "Doing Okay" for [behavior] not rate it "Doing Great!?"

   - **Most dots are in "Needs Improvement" for a particular category.** Focus on what is at the root of their struggle to achieve this effective behavior.

     What are you thinking or feeling when you are not doing [effective behavior]? What's going on with your partner or project during this time?

     What could you tell yourself or your partner to change the behavior?

     What can you do the next time you want to grab the mouse (e.g., tell your partner out loud that you really want to take over so they can explain more of what they are doing, talk about switching roles, don't grab the mouse)?

3. After discussing the key areas that stand out, commend students for sharing how they think they are doing and encourage them to keep practicing pair programming. Remind them that the most important thing is to have fun learning—not just how to program, but how to work as a team!

Chart Example

PP Behaviors	Needs improvement	Doing okay	Doing great!
[from poster]			
[from poster]			
[from poster]			
[from poster]			
[from poster]			
[from poster]			

Pair Check-In

5 minutes

Purpose

+ For students to reflect on their PP behaviors, reinforce the effective behaviors listed by the class and set goals for their partnership.

Materials

+ White board or projection screen
+ Index cards (1 per pair)
+ Markers or pencils (1 per pair)
+ **Pair Programming Roles & Responsibilities** poster

Preparation

+ Write up the following 2 questions for the whole class to see:
  - What do you think you and/or your partner are doing well?
  - How can you improve your pair programming partnership?

Activity Steps

▶ Introduction

1. Referring to **Pair Programming Roles & Responsibilities** poster, tell students to think about their answers to the posted questions. Their answers can be about one of the behaviors listed on the poster or another behavior they notice while working with their partner. Encourage students to be specific.
2. Explain to students that each partner will take a turn to verbally answer both questions.
3. Next, as a pair they will decide on ONE thing they do well and ONE thing they want to improve. They will write these behaviors down on their card with the day's date.
4. Tell students that they will keep these cards where they can see them for programming sessions and refer to them in future check-ins to remember what they are doing well and what their goals are for strengthening their programming partnership. Hand out index cards and markers/pencils to each pair.

Teacher Note

This is just one way for pairs to provide feedback and be held accountable. Rely on your own strategies for pairs or groups giving each other feedback based on what you know about your students and class. Repeat check-ins as often as you feel is beneficial for your students.

► Facilitate Activity

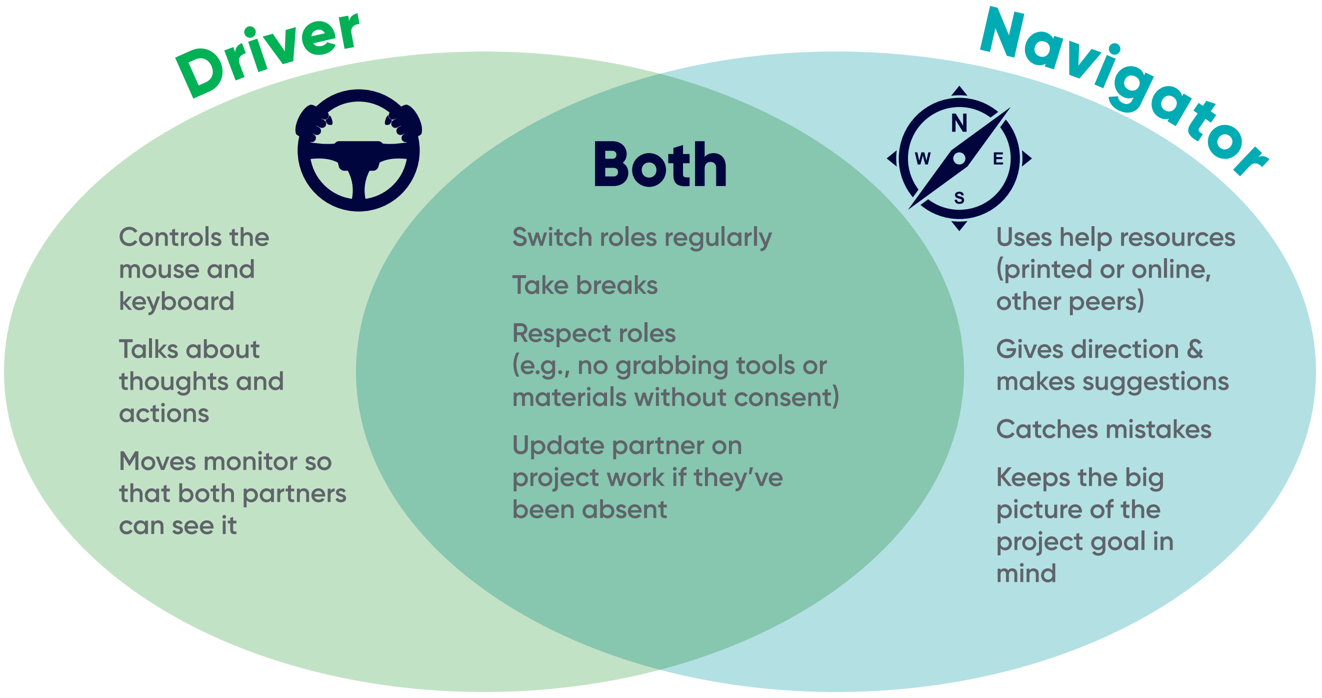
1. Hand out index cards and pencils/pens.
2. Give students a few minutes to share and write on their cards.
3. Rotate around room to encourage all pairs to write something.
4. Have the pair put the card where they can see it in a common project folder or create space for storing cards for future programming sessions and check-ins.

Teacher Note

For future check-ins, have pairs start by reviewing their past check-in cards and then make new affirmations and set new goals for improvement.

Appendix

Principles of Pair Programming Quick Reference



Driver

Controls the mouse and keyboard

Talks about thoughts and actions

Moves monitor so that both partners can see it

Both

Switch roles regularly

Take breaks

Respect roles (e.g., no grabbing tools or materials without consent)

Update partner on project work if they've been absent

Navigator

Uses help resources (printed or online, other peers)

Gives direction & makes suggestions

Catches mistakes

Keeps the big picture of the project goal in mind

Teamwork

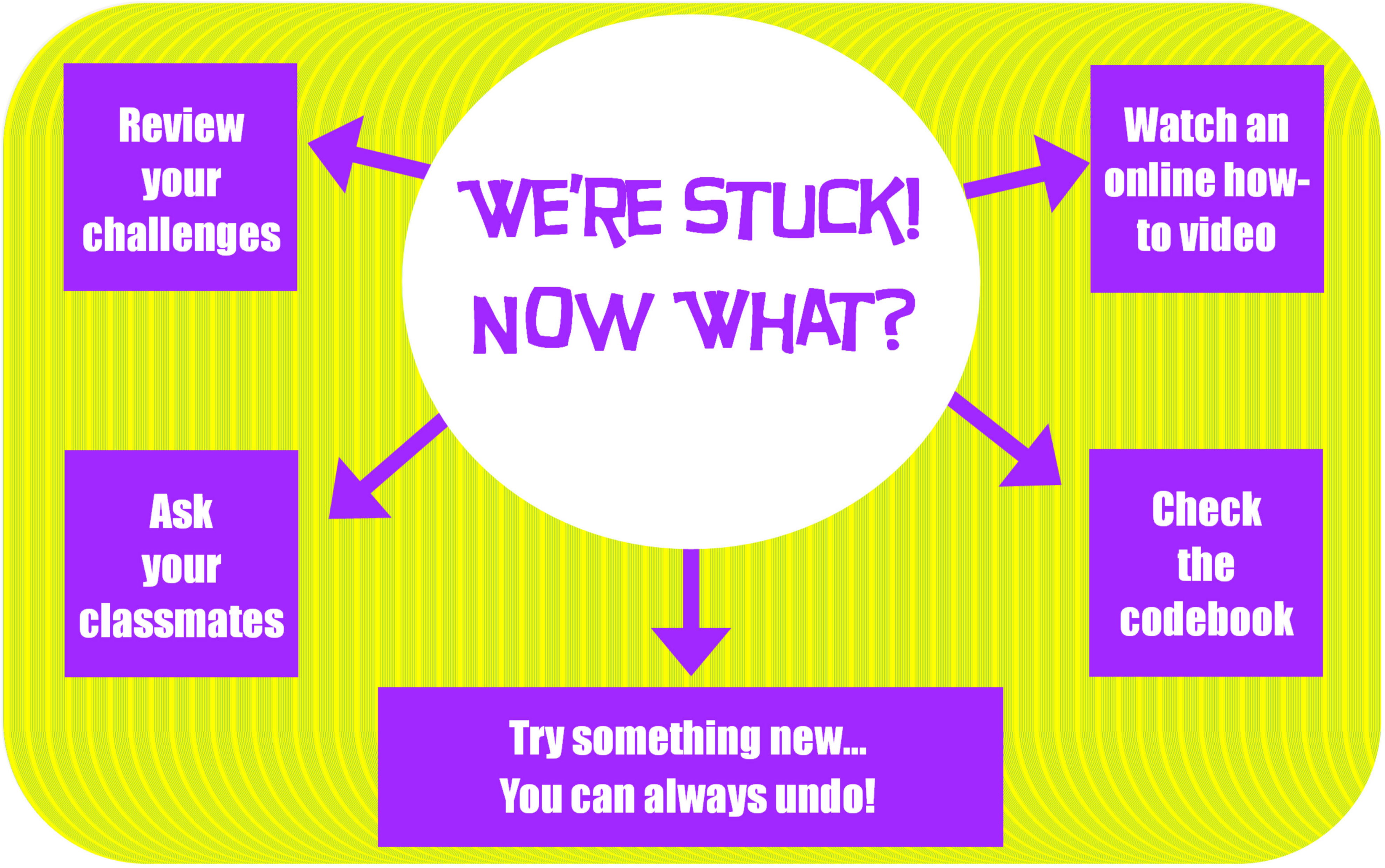
- Listen
- Compromise
- Share ideas
- Give feedback
- Check for agreement
- Respectfully disagree
- Check in about roles, engagement and programming decisions
- Step-Back sometimes if you tend to take over
- Step-In by speaking up if you tend to stay quiet

Value

- Built-in help—help each other; don't need to wait for teacher
- Can be more fun to not work alone
- 2 heads are better than 1—catch more mistakes; more design and/or problem-solving ideas
- Practice a new "coding" language while talking with your partner
- Used in industry to create a better final product
- Learn how to work on a team—helps with future school and work projects

Key Reminders

- Ground everything in class-generated **PP Roles & Responsibilities**.
- Remind pairs that there is no failing at PP—they're practicing a new skill.
- Remind pairs to switch Navigator and Driver roles to build off each others' expertise and monitor for imbalances in time or power in the Driver role.
- Affirm students for what they are doing well; include specific examples when possible.
- Use words other than "good" to describe positive PP behaviors (e.g., "effective," "beneficial," "positive").
- Emphasize teamwork and communication. Remind them to say "we" and "us" not "I" and "me."
- Remind them that "Two heads are better than one."
- Encourage more assertive students to "step-back" in the pair—to listen and respond to their partner. Encourage less assertive students to "step-in"—to talk about their ideas and help solve programming challenges.
- Challenge them to work together effectively.
- Provide suggestions of how they can improve.
- Ask students to briefly share out a couple of reasons why it's good to program with a partner.
- Encourage them to have fun creating and programming together.


WE'RE STUCK! NOW WHAT?
Review your challenges
Watch an online how-to video
Ask your classmates
Check the codebook
Try something new... You can always undo!