

Algorithms and Art

Distance Learning Support

Check with your teacher about:

- ☐ Materials or resources you need for this project
- ☐ What work you need to turn in and how to submit it
- ☐ Collaboration strategies
- ☐ If you are using a Chromebook, open a [blank code editor](#).



GOALS

- Apply all you have learned to plan and create an image of your choosing.
- Explain how a variety of algorithms work in your program.



MATERIALS

- Screen-capturing (screenshot) software

RESOURCES



Software Development Process



Brainstorming Solutions



Project Introduction

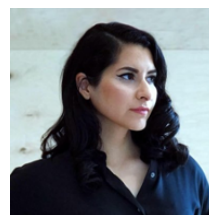
In this project, you will plan, design, and create an artistic artifact using your *Python*® programming skills to manipulate turtles. The artistic artifact you choose to create should represent the algorithmic thinking and computational practices you have learned throughout this lesson.



CAREER CONNECTIONS


Art, Design, and Technology

Danielle Forward earned a Bachelor of Fine Arts degree from the California College of the Arts and is now a Product Designer, focusing on using technology to improve connectivity and social impact. She is affiliated with the Pomo, Yokayo, and Miwok First Nations and is the founder of Natives Rising, a website dedicated to promoting the visibility of Native Americans in the technology industry and helping them achieve their goals. [Source](#)



Danielle Forward
[Source](#)

Development Process

Throughout this course, you will be given many opportunities to develop programs and other artifacts of your choice and using your own design. To help you meet your development goals, you use the **development process** . The development process for all of the projects and problems in the course will follow the same steps.



PLTW COMPUTER SCIENCE NOTEBOOK

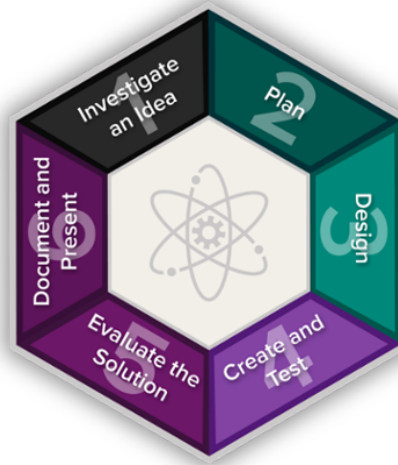
Document all project work in your PLTW Computer Science Notebook.

1

Review the [Software Development Process](#) to learn the purpose of each step of the development process.

Development Process

- Investigate an Idea
- Plan
- Design
- Create and Test
- Evaluate the Solution
- Document and Present



Throughout this project, you will be guided through each step of the process so you can create the best possible solution.

Investigate an Idea

Use the steps below to investigate an idea for your project.

2

Begin by reviewing the **requirements**  that this project must meet.

Requirements

Your artistic artifact should represent the algorithmic thinking and computational skills you have learned thus far.

Basic functionality:

- ☐ Create a variety of shapes to produce a unique, artistic artifact.
- ☐ Use color and size variations to enhance your artwork.
- ☐ Use movement to enhance your artifact.
- ☐ Use iteration (looping) and conditional execution (`if` statements) to control the drawing.

To help you in creating your artwork:

- ☐ Use existing turtle methods.
- ☐ Choose descriptive variable names.
- ☐ Comment code segments or blocks of statements.

3

Brainstorm a few ideas that you would like to create. Avoid going into too much detail, especially *how* you will create your artifact. You have probably done a fair amount of brainstorming or idea generation throughout your education. Here are some basic “rules” about brainstorming.



Basic Brainstorming Rules

1. Capture all ideas.
2. Do not evaluate and never criticize.
3. Work for quantity to allow quality ideas to surface.
4. Welcome piling on. Use other ideas as launch pads.
5. Allow a free-for-all.
6. Encourage big thinking.

Review the [Brainstorming Solutions](#) resource to learn about different brainstorming techniques.

4

Once you are done brainstorming, evaluate your ideas by choosing one or two of your ideas and elaborate on them with further documentation and rough sketches.


5



Choose one idea to fully develop.

6

Bring your idea, including documentation and sketches to your teacher for approval before continuing.


Plan

Software development projects are often organized into **milestones** , a list of tasks that need to get done by a certain date. To break down your project into smaller tasks requires problem decomposition.

Problem decomposition  is the process of breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain. In this part of the **incremental development**  process, you *decompose* your artistic artifact into smaller features and then define *milestones* for developing those features.

- 7 Decompose your problem. Identify the major parts of your program. Decide what the various parts will do. These are its features.
- 8 Decide whether each feature is *feasible*, meaning decide if it is doable. Be sure that all features are feasible and that you will be able to develop them in the time allowed. If necessary, modify your list, moving some features to an optional or “nice to have” list.
- 9 Choose what feature to develop first, what to develop last, and what to develop in between. Estimate the time you think it will take to develop each feature. These are your development milestones. Document them and assign them dates. You should have *at least* one milestone for each day you work on your project.

Design


Now that you have planned what you want to create, it is now time to **design**  its specific features.

- 10 Create sketches and/or diagrams of the major features. Before going into too much detail, review the project *requirements* and the feasibility of your design:
 - a. Project Requirements: Confirm that the feature set you are designing meets all of the requirements. Document which feature will meet each requirement. For example, which feature will demonstrate color? Movement? Iteration? Adjust your feature set if necessary and update your milestones.
 - b. Design Feasibility: Decide if any of your features need new components, new methods, or new algorithms that you

have not yet learned. Predict how long it will take you to learn any new component. Consider moving time-consuming or difficult features to the “nice to have” list. Update your milestones to reflect any changes and confirm your project still meets the requirements.

- 11 Write pseudocode for the major features of your program. This should give you a broad idea of how to develop them, but some of the details may not be completely clear.

Create and Test

In this section, you will begin to develop a **prototype** . Remember to test your prototype throughout the development.

- 12 Refer to your list of milestones and develop your first feature:
 - a. Use your pseudocode as comments, and if necessary, adjust them as you develop your solution.
 - b. Remember to use your debugging techniques when you run into problems.
 - c. Check that your feature meets the requirement(s) that you documented as part of your project milestones.
- 13 Test that the feature works and record the milestone as complete.
- 14 Make note of any challenges that you faced and how you overcame them or worked around them.
- 15 Repeat the previous steps to develop and test each feature until you have completed your project.

IMPORTANT: As you add new features, test previously developed features to ensure they work with the new additions.

16

Make note of any features you did not have time to develop and any milestones you did not reach.

Evaluate the Solution

17

Share your artistic artifact with a classmate.



Discussion Prompts

Can you read and understand your classmate's program?

- Is it well commented?
- Are variables well named?
- Is it well structured and logical?

How are iteration and conditional execution used in each of your programs?

Document and Present

A program can be described broadly by what it does, or in more detail by both what the program does and how the program statements accomplish this function. You will document your program using some of the guidelines from the College Board's Create Performance Task. The Create Performance Task requires you to describe your development process and the program you developed.



As part of your project documentation, you will take screenshots of your running program.



Screenshot Quality: Do *not* use a mobile device to take a snapshot of your screen. This results in a very poor quality image. Do use screen-capturing software available on your computer. Your teacher will tell you what software to use to capture the screenshots.

The screenshots of your artistic program should be of the same quality as the following screenshots of the “turtles in traffic” program.

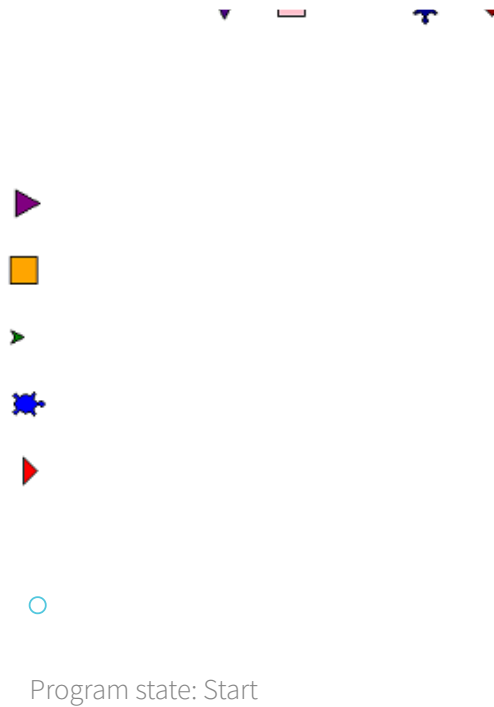




Figure 1. Turtles in Traffic Screenshots

18

Explain your program's overall functionality and how you developed it:

- a.  Create good quality screenshots of your program's output (the turtle output). You should use more than one screenshot to convey what it does.
- b. Identify the purpose of your program and explain what the screenshots illustrate.
- c. Describe two program code segments related to lists:
 - i. Show how initial data has been stored in a list.
 - ii. Show how the data in the list is being processed.Be sure your discussion identifies the name of the list being processed and what the data in the list represents.
- d. Describe one code segment related to selection. Selection determines which parts of the program executed based on a condition being true or false.

Documentation: All these things together help form **program documentation** , a written description of the function of a code segment, event, procedure, or program and how it was developed.

19

Submit your work as directed by your teacher.

CONCLUSION

- 1 Reflect on the planning and design of your program. How did it help you in your development process?

Proceed to next lesson

