PLTW COMPUTER SCIENCE

Project 1.1.6

App Development: Creative Expression

goals

- Decompose a project into smaller parts
- Apply coding fundamentals and iterative processes
- Develop an app as part of a pair programming collaboration

description of app

Create a game, an interactive story, or an app for artistic expression.

Essential Questions

1. What is the purpose of your program?
2. Where does the program integrate mathematical and/or logical concepts?
3. What does one of the algorithms do in the program?
4. How does an abstraction you created manage complexity in the program?

Essential Concepts

- Computer Science Practices, Computational Thinking, User-centered Design, Iterative Design and Testing
- Pair Programming
- Event-driven Programming
- Algorithms, Variables, Arguments, Procedures, Operators, Data Types, Logic, Loops, and Strings

Resources

- Game App
- Interactive Story App
- Interactive Media Music App
- Interpreted Performance Guide

Project Introduction

Apply what you have learned to develop an app focused on creative expression. You can choose one of the following creative development ideas:

- A game app
- An interactive story app
- An interactive media app

Examples are provided, but the intent of this project is to encourage you to use your imagination, work collaboratively with a partner, and create an app that is uniquely yours. The app is not intended to necessarily solve a problem.

PLTW DEVELOPER'S JOURNAL Document all project work in your PLTW Developer's Journal.

Problem Decomposition

Problem decomposition is an essential concept in computer science, like abstraction, algorithms, and pattern recognition. Decomposition is the process of breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain.

When people collaborate on a program, the code is often organized into pieces so that the program can be built and tested incrementally. As new features are added, the program is easier to maintain, and fixing bugs is not as hard, because teams can focus on one small part at a time. Breaking a problem into pieces is a crucial strategy for solving complex problems.

For decomposition to be a successful and powerful strategy, the components of the program must be modular. You have been creating programs modularly by looking at each individual backlog item in order and developing the item to function before you move on to the next backlog item.

Each piece of modular code should depend as little as possible on the details of other parts of the code. The pieces of code eventually will have to connect together, but the connections should be simple and well defined. That way, they can be improved or replaced without requiring all the other pieces of code to be redesigned.

During this project, your team will need to determine, How can you break the design of the program

into smaller parts that can be created and tested independently from each other? You will use the idea of decomposition to break the work into smaller chunks. Team members may even work on different parts of the program independently in parallel and then bring them together later.

Project Tips

- **File sharing in collaborations**. MIT App Inventor accounts can only be open in one place, or it might cause errors. You need to plan how to share your team .aia file at the end of each day.
- **File naming conventions**. You cannot upload an MIT App Inventor file that has the same name as another file in your account. You have been practicing naming iteration, so plan that ahead of time.
- **Component naming conventions and a common user interface design for the team**. You will want everyone to have the same user interface design so you can transfer the modular programming functions easily. You can transfer blocks through the backpack. However, it is not possible to copy the user interface design components to other projects. You may choose to all work on the user interface design as a team first, then you can share that file with all those on the team to develop the other parts of the program.
- **Can it be done in the time given?** Can you get this done in time? Is the idea worth pursuing? It is important that team members are honest about their strengths and abilities to develop the best overall project in the time provided.

Development Process

Part A: Find an Idea to Pursue

Throughout this course, you will be given opportunities to create programs around topics that interest you. So from time to time, you will be asked to take interest inventories. Consider the following questions as you and your partner decide what type of app to create.

- Did you like creating a game earlier in this lesson?
- Do you often think of ways to improve games to make them more fun?
- Do you love horses?
- Are you a good joke teller?
- Do you play sports?
- Are you involved with dance or cheerleading?
- Do you read comic books?
- Do you love music?

Creative App Choices	Examples and Applications of Coding
Game	• Sprite Smash (seen Activity 1.1.1) • Happy Balance (developed in Activity 1.1.3) • The Guessing Game (developed in Activity 1.1.4 and Activity 1.1.5)
Interactive Story	• Choose your own adventure or interactive storybook

Interactive Media	• Digital Doodle (developed in Activity 1.1.1) • Happy Accelerometer (developed in Activity 1.1.2)

1. Brainstorm ideas about what you are interested in creating. The more ideas you come up with, the better.
2. Share your brainstorm ideas with the class. Based on common interests, you will form teams according to your teacher's directions.
3. Select an idea to pursue as a team.
4. Review the information about the type of app you are going to create.

   Game App

   Interactive Story App

   Interactive Media Music App

5. Generate some more ideas with your team and what the app will attempt to do. This may include using a flowchart, storyboard, or written description with sketches.
6. Perform research and storyboard the flow of your game or performance media for teacher approval.
7. Make sure you get approval of themes, attire, songs, and story content before you start.

Part B: Document Your Development Milestones

8. Once you have explored and defined the overall idea, you will decompose the project into smaller parts or features and determine who will be responsible for which parts of the program.
9. Create and record an initial backlog.

Part C: Prepare, Investigate, and Plan

Sprints are a further breakdown of the backlog items. In previous apps, you were given the backlog, then you were given all the steps to complete each backlog item. This time, you and your team will need to determine what those steps are.

10. Select one item from the backlog that you want to accomplish first and write it at the top of the backlog. Include a clear description with an estimated time to complete. Clarify and adjust the size of the user story if necessary.
11. Decompose the user story into several sprint tasks and record the sprint tasks in a list.
12. Plan the first task, recording your plans with sentences and sketches.

    Important: It is best to design the user interface together up front.

Part D: Design, Create, and Test

13. Select a feature from the backlog and begin a sprint. Code and test iteratively, using pair programming. Record your progress at the end of every period, exporting the MIT App Inventor .aia file and saving it with a unique name.
14. At the point of each saved version, record the accomplishments and note the challenges that

you face next.

15. During your work, use each of the following strategies at least once:
    - In App Inventor, right-click a block, select **Help**, and read the documentation for the block.
    - Search for a topic in the **Google group App Inventor forum**.
16. Record the resource you used in each case, and note what you learned.

Part E: Evaluate and Reflect

17. As you complete backlog items, be sure to have someone outside your team test the app to help you identify areas of strength and potential improvement.
18. Complete a self-reflection of your app development.

Part F: Present

19. Appoint someone on the team to be responsible for bringing the final product together.
    - Review the project expectations and be able to communicate and address all items in the **Interpreted Performance Guide**.
    - Be sure to speak and dress professionally.
20. Share what your team has developed.
21. Watch the other presentations and consider:
    - What did you like about others' projects?
    - How could you improve yours?
    - Who can you talk with to add to your own skills and experience for future projects?
22. Use the Interpreted Performance Guide to evaluate your team's project and your contribution to that project.

Conclusion

1. Describe your contribution to the collaboration.
2. How did you interpret and respond to the **essential questions**? Capture your thoughts for future conversations.

Game App

Resources

AK116GameSpriteSmash116.aia (MIT App Inventor editable file)

At this point, you have already created a few games. Can you think of your own unique game you would like to create? Can you modify or expand on a previous game app?

Game apps typically rely heavily on the concept of event-driven programming.

- The user should be able to use a keyboard, mouse, and/or touch screen to affect what happens in the game.
- The game should have an objective with several milestones or achievements. A score would be sufficient, but many alternatives exist.
- The game should include progressions or levels often with increasing difficulty.

To make your game app unique, add at least one of the following:

- Increase the speed of the sprite as the game goes on.
- Decrease the size of the sprite as the user's score increases.
- Count the number of misses. (Hint: Use the when *Canvas*.*Touched* event.)
- Add another sprite as a diversion. Take points away when the user taps the wrong sprite.
- Record and display a high score that persists between games.

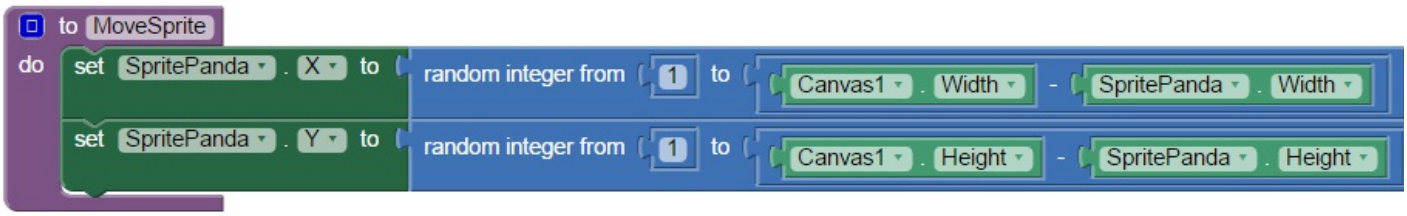
Exploring Game Apps

In the game Sprite Smash, a sprite pops up at random positions on the screen. The player scores points by tapping the sprite before it jumps to a new location, before time runs out.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Can you think of what a procedure to place the panda randomly on the screen might look like?

Explain to your elbow partner the following code in natural language (the way people talk normally). Does it make sense?



Sprite Smash is driven by an internal event (the clock timer firing) and an external event (the player tapping on the screen).

To build this game in App Inventor, apply what you have learned about programming. The example app uses a panda sprite, but you will use a sprite that matches your game theme. The sprites, sounds, and game play should all match a theme your team gets your teacher to approve.

Game App Deliverable

You may choose to create your own unique version of Sprite Smash based on the design features described above. With teacher approval, you may choose to expand on a game you created earlier in the class. Whatever game you choose to create, you will need to work with your team to create the overall mechanics of the game and then decompose the game into smaller pieces that individual team members can work on.

Important: It is a good practice to create the user interface as a team first. By doing that, the whole team will be using the same features and component names.

Interactive Story App

Resources

AK116InteractiveZs.aia (MIT App Inventor editable file)

CSE116Notes (Sample guitar chords)

Instead of a game, perhaps you prefer stories. You may bring your own story to life through an interactive story app. You can find many examples of interactive stories that let the user make choices that shape the direction of the story.

Interactive stories rely heavily on the use of conditionals to control story flow. The story should have different sections, like multiple acts in a play. For example, the story might occur on different canvases, as the user interacts.

The user inputs should fundamentally affect what happens. The story may have different endings, different dialogue, or other alternatives as approved by your teacher.

To create your unique interactive story app, you will need to:

- Create a storyboard of what actions will cause other actions in a short children's story.
- Use conditionals and logic to change values from false or true to control story flow.
- Decide whether you want to use the tablet features, such as the accelerometer, pedometer, or proximity sensor, to make the story progress.
- Use multiple images as backgrounds for canvas or button components, where the visibility of the components is controlled by the conditionals to progress the story.

Exploring Interactive Story Apps

Below is a sample user interface for the story "Interactive Zs". This interactive story app is about a boy named Zeke and his best friend Zoe. It begins with the story providing a choice. With each choice the user makes, the conditional takes the user to a different screen.

> Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Interactive Story App Deliverable

You may choose to create your own interactive story. You will need approval from your teacher, to make sure the length of your story is appropriate for the time available to create the app.

Interactive Media Music App

Resources

AK116PerformanceSong.aia (MIT App Inventor editable file)

Some apps, like Digital Doodle, are not really games or interactive stories, but fun experiences that allow users to draw, make music, or be creative. There is no score or competition, no story told, just fun artistic expression.

In this option, each person on the team will create a similar app that plays music, so that the whole team can perform a song together. Each person's app will represent a different instrument.

While some musical notes have been recorded and provided for you, you may wish to create your own unique tones. You might even connect with the choir, band, or other people with musical talents (including yourself) to record the notes that your team wants. There are two parts to creating this app:

- The first part is to trigger individual notes to play. You will use the notes to play a song and sing along.
- The second part is to program a different song to play at the push of a button. This part will use the existing sounds to play a song. To trigger the refrain, call a procedure. If the song is repetitive, like a child's song, it could also be on a counted loop, so someone could sing the song a set number of times.

Important: You need to use the *clock* component to create a delay procedure between each sound playing. Otherwise, all your tones will play over each other. You will also need a global variable *delayTime* and a *while test do* loop.

Exploring Interactive Media Apps

Review Twinkle Twinkle's user interface layout as an example:	Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

With your elbow partner explain the following code in natural language (the way people talk normally). Does it make sense?

Interactive Media Music App Deliverable

You will need approval from your teacher to make sure your song idea is appropriate. Discuss how you will record original sound files or otherwise obtain recordings of musical notes. You will need to perform the music as a team, much like a band.