Planning a programming project[1]

Becoming a programmer isn't just about learning the syntax and the concepts of a programming language: it's about figuring out how to use that knowledge to make programs. You've made a bunch of programs in your courses but now you should come up with ideas for new programs - **ideas that you're personally really excited about** - and try to turn those into actual programs.

You probably won't know everything you need for your program when you start it, and that's totally okay -- you'll be motivated to learn those new things because of how much you want to make your program real. Programmers are constantly learning new things for new projects, and that's part of why we love it so much.

Getting Started

1. Follow this link to add yourself to the CSC420 classroom and create your github repository:

   https://classroom.github.com/a/j_It6hCD

2. Once you have your repo created, get a copy of it onto your computer using git clone. It is probably a good idea to create a directory/folder for CSC420, for containing all repos related to this course.
3. Make a copy of this document
4. Select your copy of this document to edit
   a. You will edit the document using Google Documents
   b. Submission instructions at the bottom will tell you how to submit your document.
5. **ALL SOURCES THAT YOU PLAN TO CONSIDER AS PART OF YOUR PROJECT MUST BE CITED.**
6. **IT IS NOT OKAY TO SIMPLY FOLLOW A YOUTUBE OR OTHER TUTORIAL TO COMPLETE YOUR PROJECT.**
7. **THIS HAS TO BE YOUR OWN WORK.**
8. **IF IN DOUBT ABOUT SOMETHING, ASK FOR CLARIFICATION – FIRST.**

Let's step through the process of planning a programming project.

---

[1] *Planning a programming project (article) | Khan Academy*. (n.d.). Retrieved January 16, 2023, from https://www.khanacademy.org/computing/computer-programming/programming/good-practices/a/planning-a-programming-project

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

1. What do you want to make?

When I first started programming, I found myself constantly thinking of new programs to make and writing those down in a list. I was addicted to the power of creation, and there was so much my brain wanted to make. If you're like that, then you probably already have an idea of what you want to make, and perhaps you have your own list.

If you don't already have an idea, then here are some questions to help your brainstorming:

- What's your favorite game - arcade game, board game, sports game? Could you make a simplified, digital version of that? Could you mix it up a bit, like give it a different theme or main characters?
- What are your other favorite academic fields? If you love art, could you make an art-making program? If you love history, how about an interactive timeline? If you love science, how about a scientific simulation?
- What's your favorite movie or TV show? Could you make a digital version of a scene or character from it? Maybe make a game based on it?
- What's a real-life gadget that you love? Could you make a simulation of it?

Once you've picked an idea, you should write a description of it. For example, if I decided to make a clone of "Breakout", because that's my favorite retro arcade game, I might write:

Breakout: a game where you control a paddle at the bottom of the screen, and you use it to hit a ball upwards and at angles to break bricks. The goal is to break all the bricks, and not let the ball through the ground too many times.

You'll flesh that description out later, but for now, that gives you a good enough idea to keep going in the planning process.

1. **Your favorite idea from the previous assignment. Repeat the idea here. Add additional detail if you have it.**	I will create a 2D fruit ninja game where various fruits will appear on the screen and the user is responsible for clearing the screen by shooting each fruit to complete the round. At the end of the game, the user's score will be displayed and will be prompted to play again.

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

2. What technology will you use?

In this step, you need to consider which technologies (languages/libraries/environments) you're familiar with or able to learn easily, and which of them are the most well suited for the job.

You *can* learn a new technology for a new project, but especially if you're just getting started in programming, it's a good idea to get really good at your first language first.

2. Itemize the technologies you plan to use, such as languages, IDE, libraries, operating systems, etc. List all the ones you can think of.	- Lua programming language - Love2D game development Framework - Physics - Love library - Tiles - open source map design tool

Carefully read sections three (3) and four (4). Question three (3) of this assignment follows sections 3 and 4.

3. What features will it include?

This is where we get into the real planning, and where (I think) it gets fun. Your goal in this step is to figure out what you're actually making- what will it look like, what features it will include, what features it *won't* include.

The first thing you can do is make "mock-ups" – sketches that look like the thing you're making, but without details like coloring or exact sizing.

You can make mock-ups on paper, or with online programs:

To give you an idea of what mock-ups look like, I've included mock-ups below of my Breakout clone. I sketched each scene separately and drew arrows between them to show how one scene leads to another. Those arrows will help me understand what logic I need in my program to go between program states.
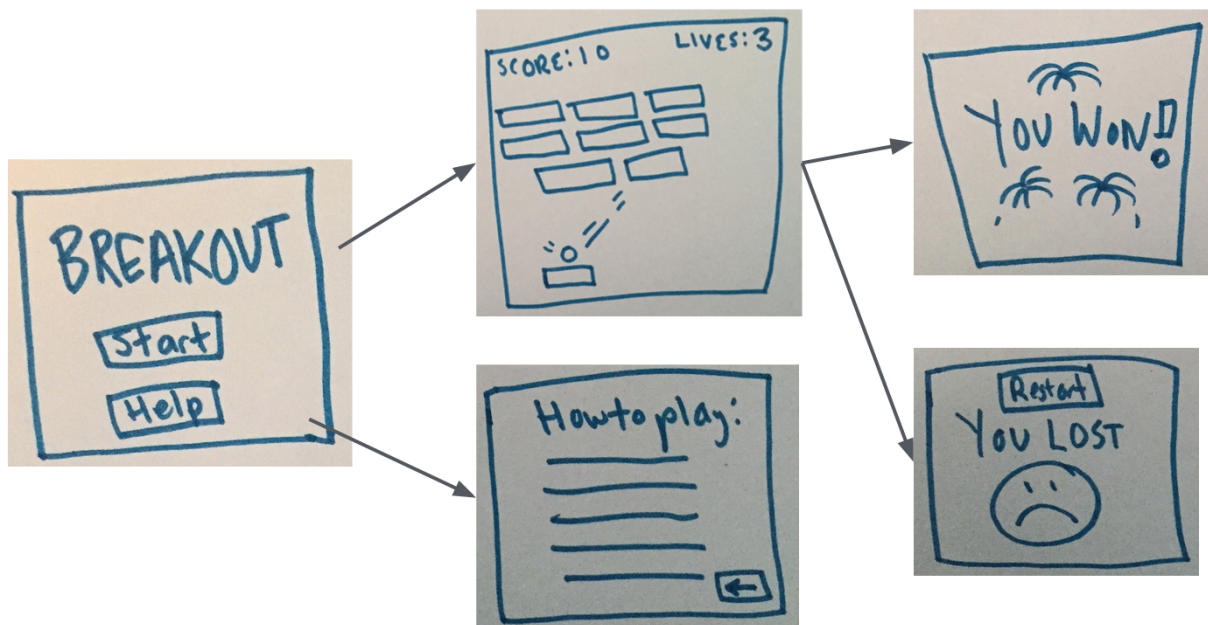
Sketched mock-ups of a Breakout clone

Now you can use those mock-ups to help you make a feature list, where you think of every feature in your program, and make it into a list.

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

CSC 420 - Programming Languages
P0127: Planning a Programming Project

For my Breakout clone, this could be my feature list, broken down by scene:

Game scene
- User-controlled paddle
- Multiple colored bricks
- Angled ball movement
- Collision detection
- Life display
- Score display
- Sound effects

Main Scene
- Play button
- Help button

Help Scene
- Text
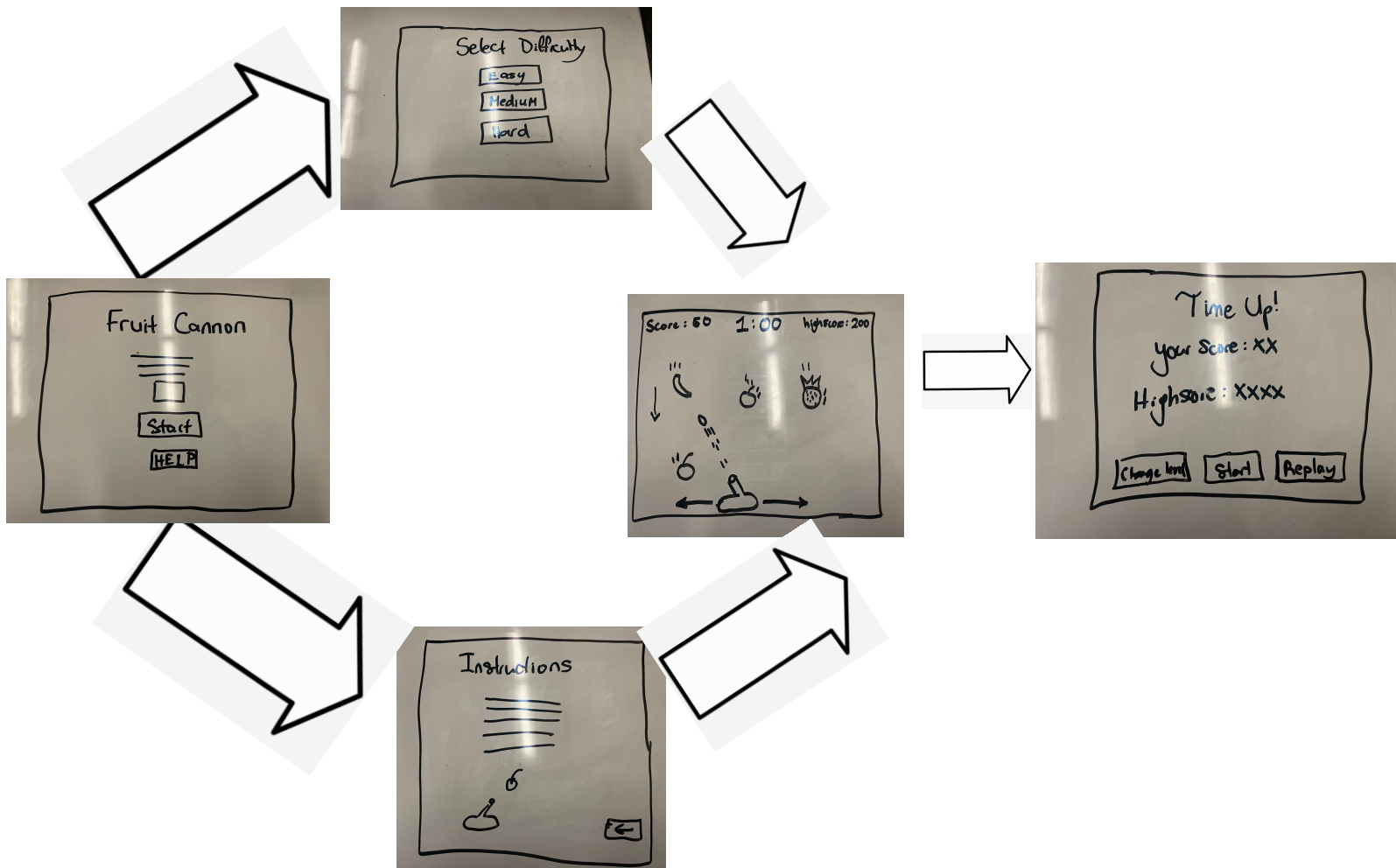- Back button

Win Scene
- Headline
- Fireworks animation

Lose Scene
- Text
- Restart button

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

Include mock-ups here

4. But what features must it include?

If we all had infinite time to make all the programs in our heads, then they'd all include every feature in our list. But we don't, so in this step, you have to decide which features are the most important, and which features you'll do only if we have time. This will also help you figure out which order to implement features in, from most to least important. To help you figure out the importance of each feature, ask yourself these questions:

●
● If I shared this with a friend, which features would I want to make sure were working?
● Which features am I the most excited about building?
● Which features are the most unique to my program?
● Which features will I learn the most from implementing?
● Are there any features that seem too far beyond my current skill level?

Then, go through your feature list from the last step, and either order the list or add a rank to each feature.

For my Breakout clone feature list, I've put "P1", "P2", and "P3" next to the features, signifying top priority (P1), middle priority (P2), and lowest priority (P3). I decided to prioritize the unique game mechanics over general game features like scenes, because I find that the most exciting about this project:

> You can also think of Priorities in these terms:
> **P1 - Must have**
> **P2 - Good to have once P1 priorities are completed**
> **P3 - Nice to have if there is time after P1 and P2 priorities are completed**

(P1) Game scene
● (P1) User-controlled paddle
● (P1) Multiple colored bricks
● (P1) Angled ball movement
● (P1) Collision detection
● (P2) Life display
● (P2) Score display
● (P3) Sound effects
(P2) Main Scene
● (P2) Play button
● (P3) Help button
(P3) Help Scene
● (P3) Text
● (P3) Back button

(P2) Win Scene
- (P2) Headline
- (P3) Fireworks animation

(P2) Lose Scene
- (P2) Text
- (P3) Restart button

As a general tip for those of you making games, here are features that I'd recommend de-prioritizing: menus, multiple levels, 3D graphics. Focus on what's unique and fun about your game, then add those extras.

You can also turn your prioritized list into project versions, so you can easily see what you need to implement in each version and you can always stop after a particular version and be happy with what you've made.

Here's what the versions would look like for my Breakout clone:

V1
- User-controlled paddle
- Multiple colored bricks
- Angled ball movement
- Collision detection

V2
- Life display
- Score display
- Start scene w/play button
- Win scene w/headline

V3
- Sound effects
- Help button
- Fireworks
- Lose scene w/Restart button

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

3. Section	Feature	Priority	Version	Notes
Game Scene	User-controllable Paddle	1	1	Example
Game Scene	Multiple colored and shaped fruits	1	1	Apple, Pineapple, Watermelon, Banana
Game Scene	Dynamic scoreboard	2	2	
Game Scene	Time countdown	2	2	0:00
Game Scene	Collision sound effects	2	2	Bang! Boom!
Game Scene	Collision graphics	3	3	
Game Scene	Collision detection	1	1	
Main Scene	Play button	1	1	
Main Scene	Help button	2	2	
Main Scene	Leaderboard	3	3	John 25 Kyle 21 Martin 19
Level Scene	Level Select	3	3	Easy / Medium / Hard
End Scene	User Score	1	1	
End Scene	High Score	2	2	
End Scene	Login	3	3	"Enter your username to save your score"
End Scene	Replay button	2	2	

Add as many rows to the table above as you need.

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

5. How will you implement it?

You now have an idea of what features you'll be building first in your program - but if you start now, you'll be staring at a completely blank program with no code written, and that can be intimidating. Which variables should you write first? Which functions?

One way you can figure that out is to think about the "high level architecture" of your program - breaking it into categories like "objects", "logic", "user interaction", "user data", and "scenes" - and then think about how you might implement them, like as object-oriented object types, functions, or variables.

For example, here's an architecture for my Breakout clone:

Objects
- `Brick(.isHit())`
- `Paddle(.move())`
- `Ball(.move())`

Scenes
- Start
- Game
- End

Logic
- Ball-brick collision (`function`, use bounding box)
- Paddle-ball angling (`function`, invert angle)

User interaction
- Keyboard-paddle movement (`keyPressed`)
- Buttons for scene changes (`mouseClicked`)

User data
- Ball deaths (`array`)
- Ball hits (`array`)

Once you've thought about the high-level architecture, it should become more-clear what you can start coding first.

You might decide to write your whole program in pseudo-code first, which we talk about later in this tutorial. Basically, it'd mean writing the whole program in plain English text inside a comment, and then slowly turning that into actual code.

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

4. Classes, Methods, User Interface/Scenes, User Data, Logic	Description
Scene: Game Scene	This scene will involve moving parts where the fruit objects will interact with the user paddle object in accordance with the game logic.
Scene: Main Scene	Introductory scene which will provide a game description as well as how to play.
Scene: End Game	This scene will be displayed once the countdown gets to 0:00 and then the user score will be listed as well as the high score. Users will be prompted to play again or exit.
Class: Fruit	Class will contain methods to move their position and methods to identify whether it has been hit or not. Various fruit objects will be called and given different attributes to change shape, size, and color.
Class: Ball	Class will contain methods to move the ball vertically along the y-axis. IsCollision methods will clear the ball from the screen once True. The speed and size of the ball will be altered as attributes.
Class: Cannon	Class will contain methods to move horizontally across the x-axis. The cannon angle will be altered as well as the speed.
User interaction: Keyboard-paddle movement (keyPressed)	The cannon direction along the x-axis will be controlled by the keyboard.
User interaction: Buttons for scene changes (mouseClicked)	User would select buttons to change the scenes
User interaction: Input username (keyPressed)	User would enter their username using the keyboard

User data: Score	Tracks the user score
Stored data: Highscore	Highest score for the game

Add additional rows to this table if you need them.

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27

6. What's your timeline?

How much time do you have to make this program? How many weeks, and how much time each day? What features will you write each week? Your goal in this step is to figure out a timeline for your project - which is particularly important if you have a deadline, but also useful so you start to understand how much time it takes you to write a program.
Here's a timeline for my Breakout clone, assuming 2-4 hours of work each week:

- Week 1: Design and pseudo-code
- Week 2: Rough visuals
- Week 3: Ball moving/collision mechanics
- Week 4: Scoring mechanics
- Week 5: Scenes (Start/Win/Lose)
- Week 6: Polish, Manual tests (QA), Prep for demo

Figuring out timelines for programming projects is hard. Some things that seem easy take way longer than you expect (like some weird bug that you spend hours debugging), some things that seem hard take less time than you expect. As a general rule, assume it will take you longer than you think, and adjust as you go along.

5. Week	Weekly Plan
1 (01/29-02/04)	Software Installation and Lua Basics
2 (02/05-02/11)	Learning Lua OOP and GIU
3 (02/12-02/18)	Design and Pseudo Code
4 (02/19-02/25)	Rough visuals
5 (02/26-03/04)	Basic logic implementation (P1 completion)
6 (03/05-03/11)	*SPRING BREAK* – WORK CAN BE DONE THIS WEEK
7 (03/12-03/13)	Project Due 2023-03-13 at 11:59 PM

Are you ready!?

Hopefully this gives you an idea for the process of planning a programming project and inspires you to start a project now.

The important thing is to make sure you start making your own programs at some point, because that is where you'll learn the most, and also where you'll get the most joy out of programming, because you're making your dreams into reality.

Submission

1. Download this document as a PDF and put it in your git repository by adding it to your repo's directory on your computer
2. git add, commit, and then push this to git by the deadline set in Moodle

Ms. Deanna M. Wilborne, MCS, CCP
Visiting Assistant Professor of Computer Science
Created: 2019-10-29; Revised: 2023-01-27