My idea

Use this to summarize your idea, plan it using sketches, notes and pseudocode as needed.

My idea is to create an arena shooter platformer arcade game.
The plan is simple: Each player starts with a default weapon – say, a pistol. The weapon aims in the direction you push the stick, and you fire it by pressing the A button. Each different weapon has different levels of recoil, bullet counts, rates of fire, reload speeds, initial ammo counts, damages, and more. Most bullets will be able to bounce off walls, ceilings, and the ground. The first player to take their opponent's health down to zero wins! Weapons, ammunition, and very rarely health packs will spawn at set locations throughout the arena. Ideally, the optimal way to play will be to control these points, thereby controlling access to powerful weapons and healing.

Things that would be nice to have (in order of most to least wanted):

- Multiple maps
- Perks each player could select before each round begins
- Multiple game modes, e.g. deathmatch (default), gun game (one kill with each weapon), capture the flag, hold the zone (one or more zones per map)
- Custom game modifications, e.g. timer, lives, recoil multiplier, item drop rate, etc.
- Different "heroes" to choose from with different stats, e.g. run speed, jump height, health, armour

Where will the inventory skills be demonstrated? List every one to be sure you've included them.

1. Line can be used to draw sliders in the menus, or for basic particle effects for explosions or bullets
2. noFill(), noStroke() will both be called at the start of the project, to reset for a clean startup
3. rectMode(CENTER) will be used pretty much immediately. It's the mode I feel most comfortable using.

4. setup() and draw() will be used once each. For obvious reasons.
5. background() will be used to set the default background colour in setup(), random() will be used for bullet spread
6. constrain() will be used to stop players from leaving the screen
7. keyPressed() and keyReleased() will be used for player input. Mouse will not be used to make the game arcade-friendly.
8. Increment operators will be used, like, everywhere I can. Any time I need to manipulate a variable in a programmatical way!
9. Local variables will be used for most complex functions.
10. Global variables will be declared before setup().

11. println() will be used very occasionally for debugging, and one will be left in the final product as an edge case exception or easter egg. I may use it in a function that would otherwise return no value if a condition is not met.

12. Conditionals will be used all over the place. For example: *if* a player's health drops below zero, end the game.
13. Boolean expressions will be used in the aforementioned conditionals.
14. Logical operators will also be used in conditionals.
15. A switch statement will be used to determine how to spawn bullets based on the gun the player is using

16. I tend to use for loops much more often than while loops. A for loop will be used to iterate over the bullets currently on screen, *for* example
17. A nested loop will almost certainly also be a part of the bullet iteration process
18. break() will be used in the case that a player's health drops to zero while in the middle of the bullet iteration process
19. (THIS WILL BE COMMITTED AS A .TXT FILE) A for loop iterates a set amount of times, while a while loop iterates for as long as a condition is met.

20. A function with no parameters or return type will be the reset() function. This will reset all player stats, such as health and current weapon, to the default values before each round begins.
21. A function with a return type will be used to calculate player position after movement, including collision detection.
22. (THIS WILL BE COMMITTED AS A .TXT FILE) Parameters are variables used when declaring a function. Arguments are the bits of data you pass through the parameters.
23. A function that takes a value as an argument will be used to draw the player at their position (player x, player y)
24. A function that takes an object as an argument will be used for collision detection, using each player object.

25. (THIS WILL BE COMMITTED AS A .TXT FILE) A class is like a cookie cutter, and an object is like the cookie it cuts. An object can be made from a class, and will inherit its properties.
26. (THIS WILL BE COMMITTED AS A .TXT FILE) A constructor function is what tells the program how to make an object. It initializes all of the values for a new object when that object is created.
27. (THIS WILL BE COMMITTED AS A .TXT FILE) Each class should have its own tab for organization purposes. Otherwise, the program gets way too long!
28. A class with a constructor function will be used to handle bullets
29. The keyword "new" will be used each time I want to spawn a new bullet
30. A constructor function with parameters will be used to create bullets – parameters may include things like damage, speed, angle, gravity, max range, etc.

31. (THIS WILL BE COMMITTED AS A .TXT FILE) An array has a predefined maximum size, whereas an ArrayList can be resized dynamically.
32. (THIS WILL BE COMMITTED AS A .TXT FILE) If you iterate through a list forwards, then deleting an item will effectively skip the next item in the list. By iterating *backwards* through the list, any deleted items only cause items you've already passed to fall back one space, and so none get skipped.
33. An array of set size will be used to store item spawn locations – both x and y
34. An ArrayList will be used to store bullet data
35. That very same ArrayList will also manage the bullet objects!

36. ArrayList.get() will be used when iterating through each bullet to get the specific bullet object required for that loop

37. (THIS WILL BE COMMITTED AS A .TXT FILE) PVector "should" be used when working with movement in two or more dimensions, as it can make calculations substantially easier.
38. The PVector class will be used to handle all player and bullet movement
39. Position, velocity, and acceleration due to gravity will be used for bullets, in particular bullets with a designated "drop"
40. Finding the direction and distance between two points will be used to calculate bullet collisions with players, and potentially for homing projectiles
41. Random 2D vectors will be used for explosion particles and juice
42. (THIS WILL BE COMMITTED AS A .TXT FILE) A normalized vector is a vector scaled so that the magnitude is 1. This is super useful for making sure your speed doesn't exceed a certain amount, or for making sure you get consistent movement regardless of whether the player is holding multiple inputs.
43. PVector.rotate() will be used for homing projectiles

48. Animation with images will be used to display players, the arena, and weapons
49. Collision detection will be used for bullet collisions with players, and for player collision with the ground, ceilings and walls

Milestone 1	Milestone 2	Milestone 3	Milestone 4
What will I deliver? I will deliver a prototype of player movement, shooting, and bullet collisions without platforms or menu.	Midpoint! I will add sprites and animation, multiple weapons, platforms, player health, and weapon requirements (ammo, maybe item pickups), as well as a basic menu setup.	You are strongly encouraged to deliver your finished game at Milestone 3. This will be the finished game. Menus will be fully added, there will hopefully be multiple arenas, and I will be able to add some polish. Playtesting should also be done by this stage!	Final deadline. If not done by this point, I'd better hustle like there's no tomorrow. If absolutely necessary, polish can be done in this phase.
Which inventory skills will this demonstrate? List them.		33, 41, 43	

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, possibly 11, 12, 13, 14, possibly 15, 16, 17, 20, possibly 21, 23, possibly 24, 28, 29, 30, 34, 35, 36, 38, possibly 39, and 40. WOW, that's a lot.	11, 15 (multiple weapons), 18, 21, 24, possibly 33, 39, possibly 43, 48, 49		
It's only really a lot since most of them are pretty simple.			
You should deliver approx. 10 skills at this milestone	You should deliver approx. 10 skills at this milestone	**You must deliver 30 inventory skills by this milestone.**	