My idea [Updated Plan]

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

This will be a top-down drifting game inspired by a mix of Japanese PS1-era drifting games (listed below), and modern 2D racing games like Vektor 2089 and Super Laser Racer. Instead of having AI opponents to race against, this will be a 'time-attack' arcade game, where the main goal is to make it from checkpoint to checkpoint without running out of time. If the player completes all 3 laps of a track without running out of time, they win the game.

The P3D extension / library will be used to allow for advanced camera movements, low-poly textured 3D models (if they match the final artstyle), and more.

This will be primarily developed for the arcade cabinets, so no mouse control will be used.

All pseudocode and sketches have been moved to the 'process' subfolder.

P3D notes (discovered through prototyping):

- Angling the camera (i.e. shifting the camera's 'eye' values away from the 'center' ones) makes all overlapping 2D primitives start z-fighting

- Blender models have to be flipped upside down (Y+ = down in processing)

- Coding a 'directionalLight' after any shape will make the shape unaffected by light
- ^^ Structure it like [GUI elements -> unaffected models -> lights -> affected models]

- spotLights? work based on vertices - a long 3D rectangle will only be lit at the 4 corners unless there's extra geometry in the middle

- Setting "frameCount = -1" resets the sketch (screen flashes white and 'setup()' is called again). All variables need to be initialized in setup() for the sketch to be re-initialized correctly

- As long as it's being updated in draw(), the Processing camera can technically render things in both orthographic and perspective mode at the same time (depending on where the code is placed)

Other game inspirations:

Side By Side Special 2000
Battle Gear 2
Wipeout
Air Race Championship
Racing Lagoon
Ridge Racer
Option Tuning Car Battle: Spec R

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

<u>Shapes</u>

1. line, ellipse, rect, triangle, quad, arc, curve
   - Particle effects and specific GUI elements (i.e. the border of the speedometer) will be created using these primitive shapes. Apart from that, most of the in-game artwork will be sprite or model-based.
2. fill, stroke, strokeWeight, noFill, noStroke, color
   - Used to set the fill / color of the GUI elements and particle effects. All primitives will likely have noStroke() used.
3. Modes: CORNER, CORNERS, CENTER, RADIUS
   - CENTER will be used when creating all primitive shapes and images.

<u>Loops</u>

16. for loop, while loop
    - Used to iterate through arrays for player cars and scenery objects before instantiating / initializing them.
17. A nested loop
    - Used when cycling through the car's boundary boxes for collision data. For example, the game checks the coordinates for each track's 'colliders' with the player car's bounding box corners.
18. break()
    - If the player's ship is currently inactive (i.e. the race is over and the 'race over' screen is transitioning), the game doesn't need to check for collisions anymore, so it will break out of the loop.
19. What's the difference between a for loop and a while loop?
    - Will be answered in a .txt file in the repo.

<u>System</u>

4. setup(), draw()
   - Setup() and draw() will be used in the main sketch file to

<u>Functions</u>

20. Declare & call a function with no parameters and no return type
    - Various functions without parameters and return types will be

initialize all of the menu & game variables, and draw() will handle in-game, frame-by-frame actions.

5. background(), random(), noise()
   - Background() will be used to colour the in-game “ground” - since the game is top-down, the ‘background’ colour will appear to be the floor.

6. constrain(), dist()
   - Constrain() will be used to keep the player’s car from rotating through corners within a predefined range, and from leaving each track’s boundaries.

7. keyPressed(), keyReleased(), keyPressed, mousePressed(), mousePressed
   - Used in the main sketch file to handle player input. Since the game is intended primarily for the arcade cabinets, the mouse will not be used.

8. increment operators: ++, +=, --, -=, *=, /=
   - Used when setting the torque / acceleration of the player vehicle, making a ‘swaying’ camera effect before every race, etc.

9. declare and use a local variable
   - Used across the program. For example, variables that help to calculate car movement / torque / positioning will be declared within the ‘car’ class.

10. declare and use a global variable
   - Used across the program. For example, variables regarding camera movement / angles will be declared in the main sketch file.

used across the program. For example, the player car will have a dedicated ‘drawcar()’ function (called by ‘draw()’ in the main sketch file) to display the player’s car to the screen.

21. Declare & call a function with a return type
   - The ‘car’ function will have a function to check if the player is currently steering the car. If they are, the function will return the current rate of steering, which will then be used to determine the rotation angle of the player’s car model. As the player turns, their car will subtly rotate around corners.

22. What’s the difference between parameters and arguments?
   - Will be answered in a .txt file in the repo.

23. Pass by copy (value): declare and use a function that takes int, float, char, etc as an argument
   - Used when loading the main game based on the

24. Pass by reference (objects): declare and use a function that takes an object as an argument
   - Used to access and adjust car parameters / stats before (or during) a race.

<u>Debugging</u>

11. println(), stop()
   - println() will be used to print debug information to the console during development (i.e. car stats & current status / position, current timer length, player parameters, etc.)

<u>Classes/objects</u>

25. What’s the difference between a class and an object?
   - Will be answered in a .txt file in the repo.

26. What is a constructor function? What does it do and when?
   - Will be answered in a .txt file in the repo.

27. Why should each class have its own tab in Processing?

- Will be answered in a .txt file in the repo.

28. Write a class with a constructor function
    - The ‘car’ class will have a constructor function to set the car’s name, stats, model file, etc.
29. Use the keyword new to instantiate an object
    - Used in the main sketch file to instantiate playable cars
30. Write a constructor function with parameters
    - Part of the ‘car’ class so that the main sketch file can instantiate separate cars

Control flow

12. conditional statements: if, else if, else
    - Used to check if the player is pressing any keys in draw() (i.e, keyPressed() sets a boolean to ‘true’, and if the boolean is true in ‘draw()’, it calls ‘carMove(left)’ or something similar).
13. Boolean expressions: ==, >=, <=, >, <, !=
    - ‘==’ is used to check what screen / menu the player is currently on. All inactive menus are disabled, and player input is only detected for each respective menu.
14. Logical operators: &&, ||
    - If the player manages to finish a race *and* beat the previous lap record, they will get an extra message on the game over screen (i.e. if (raceCompleted && (lapTime < recordTime))
15. switch statement
    - Will be used during the car & track selections screen. For example, when the player presses ‘right’, an int called ‘carSelected’ will be incremented by one. The menu will then use a switch statement to find the car that corresponds to the correct ‘carSelected’ int, and display that car’s model & stats to the screen.

Nice to Know (optional)

44. Use a timer
    - The main gameplay loop will have 2 concurrent timers - one for the current lap time (which resets every lap), and one ‘checkpoint timer’ that gets increased every time the player goes through a checkpoint. If the checkpoint timer reaches 0, the game is over.
48. Do animation with images (spritesheet or individual files)
    - The splash screen will feature animated gameplay behind the game’s logo.
49. Use collision detection between objects
    - Used to detect collisions with checkpoints and walls.

Lists

31. What’s the difference between an array and an ArrayList?

- Will be answered in a .txt file in the repo.

32. Why would you want to go through a list backwards, decrementing the index?
    - Will be answered in a .txt file in the repo.
33. Initialize and populate an array
    - Used to store background scenery (i.e. stars, flares, etc.)
34. Initialize and populate an ArrayList
    - Used to store individual tracks and cars.
35. Manage a set of objects with an array or ArrayList
    - Used to manage an array of tracks / cars.
36. Use an ArrayList method: size(), get(), remove(), contains()
    - The size() value of the array will determine how many vehicles and tracks the player can scroll through before they get returned to the first vehicle in the array.

<u>Vectors</u>

37. When should you use PVector instead of float variables?
    - Will be answered in a .txt file in the repo.
38. Use the PVector class
    - Used to handle player movement and position.
39. Do some basic physics: use position, velocity, and acceleration (due to gravity) vectors
    - Velocity and acceleration will be applied when the player controls their car.
40. Find the direction and distance between two points
    - Used to find the direction / distance between the camera’s target location vs. the actual camera location before adjusting accordingly.
41. Create a random 2D vector
    - If the player collides with certain objects, a random 2D vector will be applied to the camera for a couple of frames, giving the illusion of camera shake.
42. What is a normalized vector, why is it useful?

- Will be answered in a .txt file in the repo. 43. Using the Processing documentation look up a method in the PVector class that’s new to you and use it in your code. - random2D(), copy(), rotate();	[redacted]

Milestone 1	Milestone 2	Milestone 3	Milestone 4
What will I deliver? ~~A prototype showing a working menu system (splash screen -> car select -> track select -> controls screen -> game start).~~ ~~Working player car movement, player input, and fluid camera movement.~~ ~~“Animated” race intro (i.e. the camera starts at a close, 3D angle right beside the player’s car, before zooming out to a top-down 2D angle.~~ ~~Placeholder models and graphics.~~	**What will I deliver?** ~~All questions on the ‘skills inventory’ answered.~~ ~~Basic track layout with working checkpoints and timers.~~ ~~Arrays to hold tracks and player cars, accessible via the menus (models will be updated, but still not finalized).~~ ~~PVector data / player movement will be improved if it needs to be.~~ Placeholder music & audio. [during in-class lecture]	**You are strongly encouraged to deliver your finished game at Milestone 3.** Complete game. Particle / car thruster effects (using 2D primitives and sprites). Full collision system. Finalized menu systems with final models, sprites, animations, etc. Final music / sound effects. Camera shake. PVector data / player movement will be improved if it needs to be.	Final touches (if needed), like editing sprites, making small updates to models, etc.
Which inventory skills will this demonstrate? List them.			
~~5. background()~~	~~31. What’s the difference between an array and an ArrayList?~~	1. line, ellipse, rect, triangle, quad, arc, curve	N/A - all skills will be completed by this point.
~~7. keyPressed(), keyReleased(), etc.~~	~~32. Why would you want to go through a list backwards,~~	2. fill, stroke, strokeWeight, noFill, noStroke, color	
~~8. increment operators (+=)~~	~~37. When should you use PVector~~	3. Modes: CORNER, CORNERS,	

	~~instead of float variables?~~	CENTER, RADIUS	
~~9. declare and use a local variable~~	~~19. What's the difference between a for loop and a while loop?~~	16. for loop, while loop	
~~10. declare and use a global variable~~	~~22. What's the difference between parameters and arguments?~~	17. A nested loop	
~~11. println(), stop()~~	~~42. What is a normalized vector, why is it useful?~~	18. break()	
~~12. conditional statements: if, else if, else~~	~~25. What's the difference between a class and an object?~~	20. Declare & call a function with no parameters and no return type	
~~38. Use the PVector class~~	~~26. What is a constructor function? What does it do and when?~~	13. Boolean expressions: ==, >=, <=, >, <, !=	
~~39. Do some basic physics: use position, velocity, and acceleration vectors~~	~~27. Why should each class have its own tab in Processing?~~	14. Logical operators: &&, ||	
~~4. setup(), draw()~~	~~28. Write a class with a constructor function~~	15. switch statement	
	~~44. Use a timer~~	49. Use collision detection between objects	
	~~23. Pass by copy (value): declare and use a function that takes int, float, char, etc as an argument~~	48. Do animation with images (spritesheet or individual files)	
	~~24. Pass by reference (objects): declare and use a function that takes an object as an argument~~	40. Find the direction and distance between two points	
	~~29. Use the keyword new to instantiate an object~~	41. Create a random 2D vector	
	~~30. Write a constructor function with parameters~~	43. Using the Processing documentation look up a method in the	

		PVector class that's new to you and use it in your code.	
	33. Initialize and populate an array		
	~~34. Initialize and populate an ArrayList~~		
	~~35. Manage a set of objects with an array or ArrayList~~		
	~~36. Use an ArrayList method: size(), get(), remove(), contains()~~		
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.**	