ParkitUp -Working title

Parking Game – Vertical Slice (System-Based Design)

Core Game Idea

A low-poly arcade parking game where the player controls a car navigating compact levels filled with spatial constraints, static obstacles, dynamic hazards, and optional time pressure. The objective is precise vehicle placement inside a designated parking zone.

Core Game Systems

Gameplay Systems
Responsible for rules and mechanics.

Movement Mechanics
Defines acceleration, braking, steering, turning radius, speed limits.

Parking Rules
Defines what counts as a valid park:
– Position inside zone
– Orientation tolerance
– Velocity near zero

Collision Rules
Defines penalties or failure logic.

Level Logic
Defines obstacles, hazards, win/fail triggers.

Important Boundary
Gameplay systems decide outcomes. Nothing else touches rules.

Input Systems
Responsible for capturing player actions.

Responsibilities
– Read keyboard / controller input
– Convert raw input → commands

Example
Key pressed → “Accelerate”
Joystick tilt → “Steer Left”

State Management Systems
Responsible for tracking “what is happening”.

Tracked States
– Car position / velocity
– Collision state
– Timer state
– Level completion
– Win / Fail

Example
State = PARKING_SUCCESS
State = GAME_OVER

Feedback Systems
Responsible for communicating results.

Visual Feedback
– Brake lights
– Collision sparks
– Parking zone highlight
– Success / fail UI

Audio Feedback
– Engine sound
– Collision sound
– Success chime

Control Systems
Responsible for orchestration via game loop.

Game Loop
Input → Update → Output

Input Phase
Collect player commands.

Update Phase
Gameplay Systems:
– Apply movement
– Check collisions
– Evaluate parking rules

State Systems:
– Update timer
– Update win/fin conditions

Output Phase
Feedback Systems:
– Render visuals
– Play sounds
– Update UI

Vertical Slice Gameplay Structure

Core Player Objective
Navigate → Align → Park Precisely

Failure Conditions
– Hard collision
– Timer reaches zero (timed levels)

Win Condition
Valid parking state triggered by Gameplay System.

Levels (System-Oriented Design)

Level 1 – Core Mechanics Validation
Purpose: Validate movement + parking system.

Gameplay Focus
– Basic steering
– Static obstacles
– Generous space

Systems Stress Tested
Movement System
Parking Validation System
Collision Detection

Level 2 – Spatial Constraint Challenge
Purpose: Stress precision + collision logic.

Gameplay Focus
– Narrow paths
– Construction barriers
– Tight parking spots

Systems Stress Tested
Collision System
Steering Precision
Level Logic

Level 3 – Dynamic Hazards + Timer (Optional)
Purpose: Stress decision-making + state pressure.

Gameplay Focus
– Moving AI cars
– Strict timer

Systems Stress Tested
AI Movement Logic
Timer State System
Failure Conditions

Key Mechanics (Defined as Systems)

Driving System
Inputs → acceleration / steering
Outputs → velocity / rotation

Collision System
Inputs → physics contacts
Outputs → penalty / fail events

Parking Validation System
Inputs → car transform
Checks → bounds + orientation + velocity
Outputs → WIN event

Timer System
Inputs → delta time
Outputs → FAIL event at zero

AI Driver System (Simple)
Inputs → path nodes
Outputs → movement state

Player presses accelerate
→ Input System emits command
→ Gameplay Movement System updates velocity

→ Collision System checks contacts
→ Parking System checks zone

If valid:
→ Gameplay emits PARK_SUCCESS
→ State System updates LEVEL_COMPLETE
→ Feedback System plays success effects

Technical Scope Control (Vertical Slice Discipline)

Keep Simple
– Arcade physics
– Small levels
– Basic AI
– Minimal UI

Avoid
– Damage simulation
– Complex traffic logic
– Upgrade systems
– Open world nonsense