**Game Design Document(GDD)**

**Car Racing Game**

**1. Game Overview**

**1.1 Title**

Car Racing

**1.2 Genre**

Arcade, Racing

**1.3 Platform**

PC (Windows, Linux, Mac), potential future mobile/web versions

**1.4 Target Audience**

Casual players, racing fans, people interested in quick-paced, straightforward driving games. Suitable for ages 8 and above.

**1.5 Game Summary**

Ultimate Car Racer is an engaging 2D top-down racing game where players control a car to dodge incoming obstacle cars on a multi-lane highway. The game features a scrolling background, increasing difficulty with speed increments, and game-over mechanics on collision. Players aim to achieve high scores by surviving longer. The game combines simple controls with an exciting, fast-paced experience and retro aesthetic.

**2. Gameplay and Mechanics**

**2.1 Core Gameplay**

• The player controls a car situated in the middle-lane on a scrolling highway.

• Obstacle cars move downward, simulating forward movement.

• Players avoid obstacle cars by shifting lanes left or right using arrow keys.

• The game progressively becomes more difficult by increasing obstacle speed over time.

• The score increases with each obstacle successfully dodged.

• Game over occurs upon collision with an obstacle car.

**2.2 Player Controls**

• Left Arrow Key: Move car left (if not in the leftmost lane).

• Right Arrow Key: Move car right (if not in the rightmost lane).

• Space Bar: Restart game after game over.

**2.3 Objectives**

• Survive as long as possible while dodging obstacle cars.

• Achieve a high score by staying in the game longer.

• Test agility by maneuvering through traffic efficiently.

**2.4 Rewards & Penalties**

• Reward: Growing score based on time survived, quick reactions.

• Penalty: Collision results in game over. The car crash triggers visual and audio feedback.

**2.5 Difficulty Curve**

• Obstacles increase in speed steadily, heightening difficulty over time.

• Lane density stays constant, but obstacle speed continues to increase to challenge reflexes.

**2.6 Game Modes (Future Enhancement Ideas)**

• Endless Mode: Survive as long as possible.

• Time Trial: Complete as many dodges within a limited time.

• Challenge Mode: Introduce special obstacles or power-ups.

**3. Visual and Audio Design**

**3.1 Visual Components**

**3.1.1 Background**

• The background is a scaled image ("road\_0.png") simulating a highway.

• The background scrolls downward to simulate forward movement.

• Parallax effects (additional layers) can be added later to increase depth.

**3.1.2 Player Car**

• Positioned centrally in the lanes at (x, y) coordinates.

• Scaled to (40, 80) pixels, with a custom sprite ("car.png").

• Controlled horizontally with arrow keys, limited within lane bounds.

**3.1.3 Obstacle Cars**

• Similar in size but differing sprites ("car (2).png") to simulate different vehicles.

• Move vertically down at increasing speed.

• Randomly spawn in lanes to challenge player positioning.

**3.1.4 Crash Effect**

• A static image ("fender-bender.png") displayed briefly when collision occurs.

• Used for visual feedback of impact.

**3.1.5 UI Elements**

• Score Text: Top-left corner, updated in real-time.

• Game Over Text: Shows when the game ends, large and prominent.

• Restart Prompt: "Press Space to Restart" message.

**3.2 Color Palette**

• Bright colors for obstacle cars (e.g., red, yellow, blue).

• Track and background in darker shades to make cars stand out.

• Text in white for visibility.

**3.3 Audio Design**

3.3.1 Background Music

• Looping background sound ("car sound (2).wav") adds immersion.

**3.3.2 Sound Effects**

• Crash Sound: Play when collision occurs ("car crash sound.mp3").

• Engine or collision effects can be added later for realism.

**4. Technical Details**

**4.1 Development Environment**

• Python with Pygame library as primary tools.

• System requirements are modest, suitable for low-end computers.

**4.2 Core Components & Code Structure**

• Main Loop: Handles rendering, input, and state updates.

• Event Handling: Manages player input (arrow keys, space).

• Game States:

o START\_SCREEN - initial screen with "Play Now" button.

o ACTIVE\_GAME - gameplay is ongoing.

o GAME\_OVER - collision has happened; awaiting restart.

**4.3 Game Loop**

• Continuously checks for user input; updates obstacle positions, background scrolls.

• Increases obstacle speed over time for difficulty scaling.

• Detects collisions using Rect.colliderect() for accurate detection.

• Plays sounds and triggers visual feedback when needed.

**4.4 Performance Considerations**

• Keep obstacle count manageable.

• Use scaled sprites and static images for efficiency.

• Limit frame rate to 60 FPS.

**4.5 Extensibility & Modular Design**

• Functions for spawning obstacles, handling collisions, and resetting game.

• Future addition of power-ups or multiple levels implemented as separate modules replayable.

**5. User Interface & Experience**

**5.1 Start Screen**

• Static screen with game title and "Play Now" button.

• Clear instructions for control.

**5.2 Gameplay Screen**

• Real-time display of score, current obstacle speed.

• Smooth scrolling background creates the illusion of movement.

• Intuitive control scheme for quick reactions.

**5.3 Game Over Screen**

• Shows final score ("Game Over!") prominently.

• Re-start instructions ("Press Space to Restart").

• Optional: high score tracking, replay button.

**6. Gameplay Balancing and Difficulty**

**6.1 Speed Scaling**

• Obstacle speed starts at 5 units/step.

• Grows gradually to increase difficulty.

• Ensures balanced challenge for all skill levels.

**6.2 Lane Spawning**

• Random spawn in three lanes ([80, 180, 280]).

• Keeps game unpredictable; increases difficulty.

**6.3 Player Movement**

• Fixed movement speed (5 pixels per key press).

• No diagonal or complex movement, maintaining simplicity.

**6.4 Collision Mechanics**

• Use bounding rectangles matching sprite sizes to detect collisions reliably.

**7. Enhancements & Future Features**

**7.1 Power-Ups**

• Speed boosts, shields, or magnet effects to attract bonuses.

**7.2 Multiple Difficulty Levels**

• Auto-adjust obstacle speed or spawn rate based on player performance.

**7.3 Leaderboards & High Scores**

• Record highest scores locally or online.

**7.4 Visual & Sound Improvements**

• More detailed sprites, animated effects, particles.

• Varied background music or sound effects for richer experience.

**7.5 Additional Game Modes**

• Timed challenge mode, where player aims for maximum score within 1-minute.

**8. Art and Asset Requirements**

**Table:**

|  |  |  |
| --- | --- | --- |
| **Asset** | **Description** | **Notes** |
| Background Image | "road\_0.png" | Loopable for scrolling |
| Player Car | "car.png" | Main controlled vehicle |
| Obstacle Cars | "car (2).png" | Different models for variety |
| Crash Image | "fender-bender.png" | Display on collision |
| Sounds | "car sound (2).wav", "car crash sound.mp3" | Background and crash sounds |

**9. Testing & Optimization**

**9.1 Playtesting Goals**

• Balance difficulty progression.

• Ensure controls responsive and intuitive.

• Minimize bugs, optimize frame rate.

**9.2 Feedback Collection**

• Adjust obstacle speed, lane densities based on tester difficulty.

• Improve visual clarity based on user input.

**9.3 Future Bug Fixes & Refinements**

• Prevent bugs like overlapping obstacles or sprite flickering.

• Add visual cues for upcoming obstacles.

**10. Conclusion**

Ultimate Car Racer offers an exciting, accessible driving game experience emphasizing quick reflexes and timing. Its simple control scheme and visually clear UI make it appealing to all ages. With planned enhancements like power-ups, new modes, and social features, the game can evolve into a richer, highly replayable experience.