VANTAGE - Arcade Car Racing Game Software Requirements Specification (SRS)

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive overview of the design and functionality of the VANTAGE Arcade Car Racing Game.

1.2 Scope

This document outlines the features, modules, and requirements of the VANTAGE game, detailing its architecture, external interfaces, and functional specifications. It serves as a guide for developers, testers, and stakeholders involved in the project.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- FPS: Frames Per Second

1.4 References

- Pygame Documentation
- GNU Public License

1.5 Overview

VANTAGE is an arcade-style car racing game designed to provide an engaging and immersive experience for players. The game includes features such as player selection, multiple levels, competitors, tunnels, and various in-game objects. The document covers the overall system architecture, external interfaces, functional requirements, performance requirements, design constraints, software system attributes, and other relevant aspects.

2. System Overview

2.1 System Architecture

The VANTAGE game follows a modular architecture with key components including player selection, title screen, game flow management, level generation, and rendering. These components interact to create a seamless and enjoyable gaming experience.

2.2 Directory Structure

The game directory is organized into subdirectories, such as 'build,' 'documentations,' 'scripts,' 'lib,' 'dat,' and 'vintage.' Each subdirectory serves a specific purpose, such as housing executable files, documentation, scripts for level generation, game libraries, high scores data, and the main game functionalities.

2.3 Modules Overview

The game is divided into several modules, each responsible for specific aspects of the game. Notable modules include 'background,' 'competitor,' 'countdown,' 'credits,' 'game,' 'high_scores,' 'level,' 'main,' 'player_select,' 'player,' 'segment,' 'settings,' 'sprite,' 'title_screen,' 'tunnel_entrance,' 'tunnel_inside,' and 'world_object.'

3. External Interface Requirements

3.1 User Interfaces

3.1.1 Game Window

The game window provides the main interface for gameplay. It renders the game environment, player's car, competitors, and other in-game elements.

3.1.2 Player Selection Screen

The player selection screen allows users to navigate through available characters, view details, and choose a player for the game.

3.1.3 Title Screen

The title screen serves as an introduction to the game, displaying logos, animations, and waiting for user input to proceed.

3.2 Hardware Interfaces

The game interacts with the hardware components for user input, such as keyboard or controller devices.

3.3 Software Interfaces

The game utilizes the Pygame library for graphics, sound, and event handling.

3.4 Communication Interfaces

No external communication interfaces are required for the standalone game.

4. Functional Requirements

4.1 Player Selection

4.1.1 Navigate Player Selection Screen

Players can navigate through available characters using input devices to choose a character.

4.1.2 Choose Player

Players can finalize their selection, confirming the chosen player for the game.

4.2 Title Screen

4.2.1 Display Title Screen

The title screen displays logos and animations to introduce the game.

4.2.2 Wait for User Input

The title screen waits for user input, allowing players to proceed to the player selection screen.

4.3 Game Flow

4.3.1 Initialize Game

The game initializes with the selected player, setting up the game environment.

4.3.2 Play Game

Players control their car, competing against competitors, and completing laps in the race.

4.3.3 Pause Game

Players can pause the game during gameplay, displaying a 'Paused' message.

4.3.4 Display High Scores

High scores are displayed between levels or when waiting for a new player.

4.3.5 Handle Game Over

The game handles the end of a player's session, displaying game over information.

4.3.6 Handle Level Completion

Upon completing a level, the game displays relevant information and progresses to the next level.

4.4 Level Generation

4.4.1 Generate Gold Coast Level

The game generates the Gold Coast level with specific track segments and features.

4.4.2 Generate Melbourne Level

The game generates the Melbourne level with unique track segments and features.

4.4.3 Generate Test Level

A test level is generated with various features for testing purposes.

4.5 Rendering

4.5.1 Render Background

The background is rendered, providing a scrollable environment during gameplay.

4.5.2 Render Competitors

Competitor cars are rendered on the track, providing a dynamic racing experience.

4.5.3 Render Countdown

A countdown is displayed before the start of a level.

4.5.4 Render Credits

Credits are displayed at the end of the game.

4.5.5 Render Game

The main game environment is rendered, including the player's car and track.

4.5.6 Render High Scores

High scores are rendered for players to view.

4.5.7 Render Player

The player's car is rendered, responding to user input for steering and acceleration.

4.5.8 Render Sprites

Various sprites, including bonuses, speed boosts, and competitors, are rendered on the track.

4.5.9 Render Tunnel Entrance

The tunnel entrance is rendered as the player approaches.

4.5.10 Render Tunnel Inside

The interior of the tunnel, including walls and roof, is rendered as the player passes through.

5. Performance Requirements

5.1 Frame Rate

The game should maintain a smooth frame rate to ensure a visually appealing and responsive gaming experience. The target frame rate is set to 60 FPS.

5.2 Load Time

The game should load within a reasonable time frame, providing quick access to gameplay. The load time target is set to be under 10 seconds.

6. Design Constraints

6.1 Pygame Library

The game is designed to utilize the Pygame library for graphics, sound, and event handling. Any changes or updates to Pygame may impact the game's functionality.

6.2 Operating System Compatibility

The game is designed to run on platforms compatible with Pygame. Compatibility with future operating systems or changes to existing operating systems should be considered.

6.3 Hardware Requirements

Players need compatible input devices, such as a keyboard or controller, to interact with the game. The game's performance may vary based on the hardware specifications of the player's system.

7. Software System Attributes

7.1 Reliability

The game should be reliable, providing a stable and consistent experience without unexpected crashes or errors.

7.2 Availability

The game should be available for players to access and play without unnecessary downtime.

7.3 Security

As a standalone game, security considerations primarily involve protecting user data and preventing unauthorized access.

7.4 Maintainability

The game should be designed with maintainability in mind, allowing for future updates, enhancements, and bug fixes.

7.5 Portability

The game, built on the Pygame library, should be portable across platforms compatible with Pygame.

8. Other Requirements

8.1 Licensing

The game is licensed under the GNU Public License. All licensing requirements and attributions must be adhered to.

8.2 Documentation

Comprehensive documentation, including this Software Requirements Specification, should be maintained for reference by developers, testers, and other stakeholders.

8.3 Testing

A thorough testing process should be conducted to ensure the game's functionality, performance, and reliability. Testing should cover player selection, gameplay mechanics, level generation, rendering, and other critical aspects.

8.4 Usability

The game should provide a user-friendly experience, with intuitive controls and clear instructions. The player interface should be designed for ease of navigation and understanding.

8.5 Error Handling

The game should incorporate robust error handling mechanisms to gracefully manage unexpected situations, providing informative error messages and avoiding crashes.