Software Requirements Specification

for

Angry Birds Alike

Author: Al Jubair Hossain

Date: March 31, 2024

Revision History

Date	Version	Notes
Feb 3, 2024	1.0	Initial draft completed.
February 26, 2024	1.5	A fixed version with sections of 1.4, 4.1, 4.2, 5 were modified with specific information of physics computations.
March 31, 2024	2.0	Final draft was completed with proposing methods for calculation ,and changes in all the sections of 3, 4.1-4.5.5, 5 ,9, 10 , 12 , 13, 16 ,17,19,20, 21 ,22, Appendix with specific info that was proposed in the project. Also, added the table of contents correctly and then adding all the formula and methods incorporating with the project .

Contents

1 Reference Material

This section records information for easy reference, providing a foundation for the document's contents and ensuring a common understanding of terms and units used throughout.

1.1 Table of Units

Given the focus on physics simulations in the game, the following units are primarily used:

Symbol	Unit	Description
m	Meter	Unit of length, significant for distance calculations in simulat
kg	Kilogram	Unit of mass, crucial for force and momentum calculations
s	Second	Unit of time, used for time-stepping in physics simulations
m/s	Meter per Second	Unit of velocity, describing the speed of objects
m/s^2	Meter per Second Squared	Unit of acceleration, including gravity
N	Newton	Unit of force, applicable in simulations involving collisions
rad	Radian	Unit for angular measurements, used in trajectory angles

Table 2: Primary Units used in the document

1.2 Table of Symbols

This document uses various symbols to describe physical quantities:

Symbol	Unit	Description
v	m/s	Velocity
a	m/s^2	Acceleration
F	N	Force
m	kg	Mass
g	m/s^2	Acceleration due to gravity

Table 3: Symbols and their descriptions

1.3 Abbreviations and Acronyms

The document utilizes various abbreviations and acronyms to streamline discussion:

Abbreviation/Acronym	Meaning
SRS	Software Requirements Specification
RK4	Runge-Kutta 4th Order Method
SI	International System of Units
AB_Sim	Angry Birds Alike Simulation

Table 4: Abbreviations and acronyms

1.4 Mathematical Notation

The document employs mathematical notation consistent with physics and engineering principles, where: - Scalars (e.g., mass, time) are denoted in italic (e.g., m, t). - Vectors (e.g., velocity, force) are represented in bold (e.g., $\mathbf{v}$, $\mathbf{F}$). - Differential equations and formulas integral to the game's physics engine are presented in standard mathematical format, facilitating clear communication of algorithms and methods, such as the Runge-Kutta 4 (RK4) method for numerical integration.

2 Introduction

2.1 Purpose of Document

This Software Requirements Specification (SRS) document delineates the comprehensive specifications for "Angry Birds Alike," a physics-based game simulation designed to offer both an engaging gaming experience and an educational platform for physics students and enthusiasts. The document serves as a foundational blueprint for the development team, outlining the game's intended functionalities, performance expectations, and user interaction paradigms. Additionally, it acts as a reference point for stakeholders to understand the project's scope, objectives, and technical constraints, facilitating a shared vision for the project's successful completion.

2.2 Scope of Requirements

"Angry Birds Alike" aspires to transcend traditional gaming boundaries by integrating rigorous physics simulations that mirror real-world dynamics. The game's scope includes the design and development of a physics engine capable of simulating force, trajectory, momentum transfer, and elastic collisions. Coupled with an intuitive user interface, the game aims to challenge players to apply principles of physics to navigate through levels creatively. This document captures all requisite features, from user interactions, graphical representations, physics calculations, to the system's performance and scalability. It encompasses both functional and non-functional requirements essential for delivering a comprehensive and immersive gaming and learning experience.

2.3 Characteristics of Intended Reader

The intended audience for this document encompasses a broad spectrum of readers, including software developers, game designers, project managers, educational content creators, and stakeholders with vested interests in educational technology. A foundational understanding of software development processes, basic physics concepts, and game design principles is assumed. The document is structured to be accessible to readers with varying levels of technical expertise, providing both high-level overviews and detailed technical specifications.

2.4 Organization of Document

This document is organized into several key sections to facilitate ease of navigation and comprehension:
- **General System Description:** Outlines the high-level overview of the game, including its context, user characteristics, and system constraints.
- **Specific System Description:** Delves into the detailed problem statement, theoretical models, data definitions, and solution characteristics.
- **System Features and Requirements:** Enumerates the game's core functionalities, performance metrics, software attributes, and both functional and non-functional requirements.
- **Development Plan:** Provides a roadmap for the game's development lifecycle, including phases from initial requirements gathering to post-launch support.

- **Appendices:** Offers additional resources, such as a glossary of terms, user stories, compliance requirements, and detailed explanations of complex concepts like the RK4 method.

3 General System Description

This section provides a broad overview of the "Angry Birds Alike" system, detailing its main components, interfaces, user interactions, and operational constraints. It sets the stage for understanding the game's purpose, its interaction with users, and the environment in which it operates.

3.1 System Context

"Angry Birds Alike" is designed as a standalone application that engages users in physics-based puzzles. The system interacts with users through a graphical user interface (GUI), where players input actions via mouse or touchscreen gestures. The application processes these inputs to simulate and display physics-based trajectories and collisions. Data, such as level progress, scores, and user settings, are stored locally on the user's device. The game does not require an internet connection for core gameplay but may connect to online servers for updates, leaderboards, and sharing achievements.

3.2 User Characteristics

The primary users of "Angry Birds Alike" include individuals with an interest in physics, puzzle games, and educational software. The game is designed to be accessible to users of varying ages and educational backgrounds, from elementary students learning basic physics to adults seeking challenging puzzles. No specific technical skills are required to play the game, but users with a basic understanding of physics concepts may have an initial advantage.

3.3 System Constraints and Limitations

The game is developed to run on multiple platforms, including Windows, macOS, Android, and iOS, with considerations for the varying processing power and screen sizes. It is built with scalability in mind, allowing for the addition of new levels, physics simulations, and user interfaces without significant overhauls. However, the complexity of physics simulations is balanced with the need for the game to run smoothly on lower-end devices, which may limit the simulation's detail or the number of simultaneous interactions.

3.4 Assumptions and Dependencies

It is assumed that users have access to a device capable of running the game with a modern operating system. The game's performance and features depend on the device's hardware capabilities, such as CPU speed, graphics processing power, and available memory. Future versions of the game may depend on updates to third-party libraries such as SFML and Boost.odeint for graphical rendering and physics simulations.

4 Specific System Description

4.1 Problem Description

"Angry Birds Alike" simulates a vibrant, physics-based environment where players utilize principles of classical mechanics to solve puzzles. The challenge involves using a slingshot to launch birds with specific properties at structures, aiming to achieve maximum impact with strategic force and angle decisions. This scenario serves as an educational platform, promoting an intuitive understanding of physics concepts like projectile motion, elasticity, and conservation of energy.

4.2 Terminology and Definitions

This section introduces key terms that are essential for understanding the game's mechanics and physics simulations:

4.3 Physics Engine and Game Mechanics

The game's physics engine is designed to simulate realistic interactions based on classical mechanics principles. Utilizing the Runge-Kutta 4th Order Method (RK4) for numerical integration, the engine accurately predicts projectile trajectories, collision outcomes, and object movements under gravitational forces.

4.3.1 Theoretical Background

The engine incorporates several key physics concepts:

- Projectile Motion: Calculated using v_0, launch angle θ, and gravitational acceleration g, to determine trajectory.
- Collisions: Simulates elastic and inelastic collisions, applying conservation of momentum and energy principles.
- Gravity Effects: All objects are subjected to gravitational forces, influencing their motion.

4.3.2 Implementation Details

The physics engine is implemented in C++ with the Boost.odeint library for solving differential equations. Key formulas include:

$$\frac{dy}{dt} = v \tag{1}$$

$$\frac{dv}{dt} = -g \tag{2}$$

$$y(t) = y_0 + v_0 t + \frac{1}{2} g t^2 \tag{3}$$

These equations form the basis for calculating object positions over time.

Projectile Motion: Describes the motion of an object thrown into space upon which external forces, such as gravity, act.

Elastic Collision: A type of collision where the total kinetic energy of the colliding bodies after interaction is equal to their kinetic energy before interaction, ideal for understanding energy conservation.

Momentum: A measure of the motion of a body, equal to the product of its mass and velocity. Momentum conservation is a fundamental concept in collisions.

Runge-Kutta 4 Method (RK4): A numerical method used to approximate solutions to ordinary differential equations, critical for simulating precise projectile trajectories and other physics phenomena in the game.

4.4 Physical System Description

The game world comprises the following physical elements, each interacting according to the laws of physics:
Projectiles (Birds): Characters with unique masses and sizes, used by players to hit targets. Targets (Pigs): Objects to be hit by the projectiles, often protected by various structures. Obstacles: Various structures built from materials with different properties (wood, ice, stone) affecting the projectile's trajectory and collision outcomes.

4.5 Goal Statements

The primary goals of "Angry Birds Alike" include:
GS1: To provide a realistic physics simulation that allows players to apply and observe principles of projectile motion, force, and momentum in a fun and engaging way. GS2: To offer a challenging puzzle game where players must use strategic thinking and a basic understanding of physics to progress through levels. GS3: To create an educational tool that enhances the player's understanding of physics through interactive gameplay and immediate feedback on the physical consequences of their actions.

4.6 Solution Characteristics Specification

4.6.1 Assumptions

The game's physics engine can simulate real-world physics with high accuracy within computational constraints. Players possess or will develop a basic understanding of physics principles through gameplay. The user's device has sufficient processing power to handle real-time physics calculations and graphics rendering.

4.6.2 Theoretical Models

The game's physics engine integrates several theoretical models to simulate realistic interactions:
Projectile Motion Model (TM1): Determines the trajectory of birds based on initial force and angle, incorporating air resistance at higher levels. Elastic Collision Model (TM2): Calculates energy and momentum transfer during collisions, providing the basis for interactions between birds, pigs, and obstacles. Gravity Model (TM3): Applies a constant acceleration downwards, affecting all non-static objects in the game environment.

4.6.3 Data Definitions

Key data elements used in the game's physics calculations include:
Initial Conditions (DD1): Position, velocity, and mass of projectiles at the start of their trajectory. Material Properties (DD2): Density, elasticity, and friction coefficients for different obstacle materials, influencing collision outcomes.

4.6.4 Instance Models

Specific instances where theoretical models are applied to simulate game scenarios: Trajectory Calculation (IM1): Utilizes TM1 to predict and animate the flight path of projectiles in real-time. Collision Detection and Response (IM2): Employs TM2 and TM3 to detect when and how projectiles interact with targets and obstacles, calculating the outcome based on physics principles like conservation of momentum and energy transfer.

4.6.5 Environment Interaction (IM3):

Factors in external influences such as wind or gravitational variations in special levels, adjusting projectile trajectories and collision impacts accordingly, introducing complex scenarios for players to strategize around.

5 System Features

This section outlines the major features of "Angry Birds Alike", detailing how each contributes to the gameplay experience and educational value of the game. These features are designed to engage players in physics concepts through interactive and challenging gameplay.

5.1 Projectile Simulation

Description: This feature allows players to control the angle and velocity of projectiles (birds) launched at targets (pigs). The simulation includes real-world physics considerations such as gravity, drag, and the material properties of obstacles.
Functional Requirements:

- Players must be able to adjust the launch angle and force applied to the projectile.
- The game will calculate the projectile's trajectory using the Runge-Kutta 4 method for accurate physics simulation.
- Collisions with targets and obstacles must reflect realistic physics interactions.

5.2 Level Design and Physics Challenges

Description: Levels are designed to introduce players to various physics concepts, including projectile motion, elasticity, and momentum conservation. Each level presents unique challenges that require players to apply physics principles to solve.
Functional Requirements:

- Levels progressively increase in difficulty to match the player's understanding and skills.
- The game provides visual and textual hints related to the physics concepts explored in each level.
- Success criteria for levels are based on physics objectives, such as knocking down structures or reaching specific targets.

5.3 Interactive Physics Sandbox

Description: An open-ended mode where players can experiment with different physics setups. This sandbox feature serves as both a playground and a learning tool, enabling players to explore physics principles without the constraints of structured levels.
Functional Requirements:

- Provide a variety of objects, materials, and environmental conditions for players to experiment with.
- Allow players to adjust variables such as gravity, air resistance, and object properties.
- Include tools for measuring and displaying velocities, forces, and trajectories to facilitate learning.

5.4 Educational Content and Tutorials

Description: To enhance the game's educational value, this feature includes tutorials and in-game explanations of the physics principles at play. It aims to bridge the gap between gameplay and learning, making complex concepts accessible and engaging.
Functional Requirements:

- Tutorials must clearly explain how to play the game and how physics is integrated into the gameplay.
- In-game content should include explanations of physics concepts, examples, and real-world applications.
- The game should offer feedback on the player's actions, highlighting the physics principles involved in their successes and failures.

6 Performance Requirements

- The game shall maintain a high frame rate for smooth animations during simulations.
- Physics calculations for projectile motion and collisions shall be computed in real-time to provide immediate feedback to player actions.

7 Software System Attributes

7.1 Reliability

The game shall have error handling for user inputs and unexpected behaviors to prevent crashes and ensure

8 Software System Attributes

8.1 Reliability

The game shall have error handling for user inputs and unexpected behaviors to prevent crashes and ensure consistent performance across various platforms.

8.2 Maintainability

Code shall be well-documented and modular to facilitate easy updates and incorporation of new physics models or game features.

8.3 Portability

The game will be developed to run on various operating systems including Windows, macOS, and Linux without requiring significant changes to the codebase.

9 Technological Overview

9.1 SFML

SFML provides a simple interface to the various components of a PC, simplifying the game development process and allowing for a focus on the game mechanics and physics.

9.2 Boost.odeint

Utilized for its advanced ODE solving capabilities, allowing for accurate and efficient simulation of the game's physics.

9.3 C++ Standard Library

Used for its robust performance and reliability, particularly the data structures, mathematical functions, and threading capabilities for managing the game's physics simulations.

10 Physics Simulation Detail

10.1 Projectile Simulation

10.1.1 Runge-Kutta Method

We use RK4 for trajectory prediction by numerically integrating the equations of motion. The game updates the state of the projectile based on the forces applied, such as the initial throw and gravity.

10.2 Collision Detection and Response

10.2.1 Elastic Collision

When a collision is detected between two entities, the game calculates the resulting velocities and direction changes according to the principles of elastic collision.

10.2.2 Momentum Transfer

Momentum conservation is applied to all collisions within the game to simulate realistic interactions between objects.

11 Challenges and Improvements

11.1 Challenges

- Implementation of the RK4 method for accurate physics simulation.
- Creation of a robust stepper function for physics updates.
- Accurate momentum transfer in multi-object collisions.
- Efficient collision detection among multiple targets.

11.2 Improvements

- Introduction of advanced computational logic such as piecewise functions for enhanced gameplay mechanics.
- Inclusion of additional physics elements such as wind forces and air resistance.
- Enhanced visual effects for a more immersive gaming experience.
- Addition of new levels and features, including time-limited challenges to increase game complexity.

12 Functional Requirements

The game must satisfy the following functional requirements:

1. Provide accurate simulations of projectile motion using the RK4 method.
2. Allow players to interactively control the angle and force of projectile launch.
3. Simulate physics in real-time to provide immediate feedback.
4. Include multiple levels with varying physics challenges.
5. Implement a scoring system based on the efficiency and accuracy of hitting targets.

13 Non-functional Requirements

The non-functional requirements of the game include:

1. **Usability:** The game should be easy to learn but challenging to master.
2. **Performance:** The game should run smoothly on all supported platforms with minimal loading times.
3. **Accessibility:** The game should be accessible, following the WCAG guidelines where applicable.

14 Traceability Matrices and Graphs

This section contains the traceability matrices that map system features to their corresponding requirements, design components, and instance models, ensuring comprehensive coverage and verification.

14.1 Requirement to Instance Model Traceability Matrix

This matrix correlates each requirement with the instance models and design elements that implement them.

Req ID	**Requirement Description**	**Instance Model(s)**	**Design Element(s)**
FR1	Accurate projectile motion simulation	IM1	Physics Engine
FR2	Interactive control for projectile launch	IM1	User Interface Module
FR3	Real-time physics feedback during gameplay	IM1, IM2, IM3	Simulation Module
FR4	Multiple levels with varied physics challenges	IM1, IM2	Level Design System
FR5	Scoring system reflecting physics accuracy	IM2	Scoring Algorithm

NFR1	Ease of learning and gameplay challenge	–	Tutorial System
NFR2	Smooth operation on supported platforms	–	Performance Optimization
NFR3	Accessibility following WCAG guidelines	–	Accessibility Features

14.2 Design to Test Case Traceability Matrix

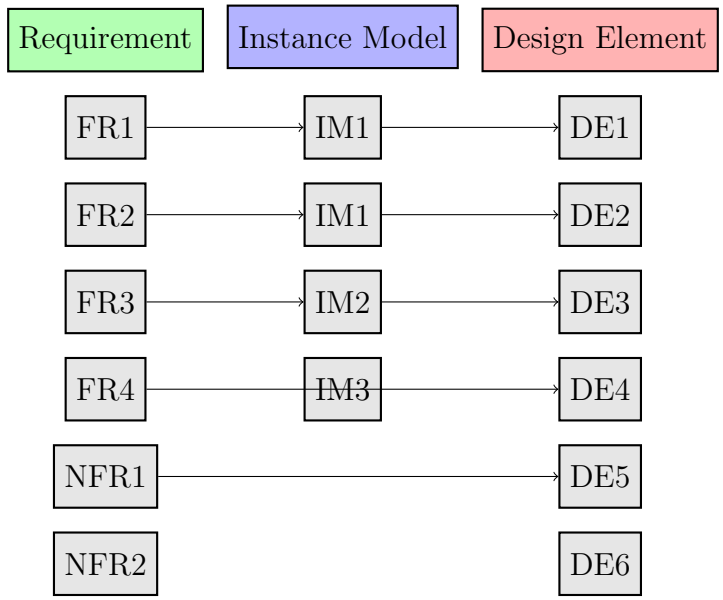
This matrix ensures that all design elements are covered by the test cases.

Design ID	Design Element Description	Test Case(s)
DE1	Physics Engine for simulating motion	TC01, TC02
DE2	User Interface or inputs for interaction	TC03
DE3	Simulation Module for gameplay feedback	TC04, TC05
DE4	Level Design System for various challenges	TC06
DE5	Scoring Algorithm for accuracy	TC07
DE6	Tutorial System for ease of learning	TC08
DE7	Performance Optimization for smooth operation	TC09
DE8	Accessibility Features for compliance	TC10

Note: Test cases (TC) are not explicitly mentioned in the provided document. They should be detailed in the test plan or quality assurance documentation and then linked to this matrix.

14.3 Design to Test Case Traceability Matrix Graph

This graph ensures that all design element correlates with all the Traceability models.



15 Enhanced System Description

Angry Birds Alike, inspired by the dynamics of "Angry Birds Game", is a sophisticated physics engine designed for educational simulations. It focuses on demonstrating intermediate to advanced physics mechanisms such as force, velocity, elastic collision, and momentum transfer. The engine utilizes mathematical formulas, specifically the Runge-Kutta 4 (RK4) method, to solve projectile trajectories accurately.

15.1 Overview of Technologies

- **SFML (Simple and Fast Multimedia Library):** Utilized for window management, graphics rendering, and event handling.
- **Boost.odeint:** A comprehensive C++ library employed for solving ordinary differential equations (ODEs), crucial for simulating the physics interactions within the game.
- **C++ Standard Library Features:** Including Arrays for data management, cmath for mathematical calculations, chrono for managing time, and thread for optimizing performance through multi-threading.

16 Physics Simulation Walkthrough

The core of ProgName's simulation is built on accurately determining the trajectory of projectiles under various forces. This is achieved through the RK4 method, allowing for precise calculation of distances covered by the projectiles.

16.1 RK4 Method Implementation

Given the differential equations:

$$\frac{dx}{dt} = vx, \quad \frac{dy}{dt} = vy, \quad \frac{dvx}{dt} = 0, \quad \frac{dvy}{dt} = g$$

where g is the acceleration due to gravity. With initial conditions at t_0 set as $x_0 = 0$, $y_0 = 0$, $vx_0 = 10m/s$, and $vy_0 = 10m/s$, we find the state after a timestep $\Delta t = 1s$, applying the RK4 method.

16.1.1 Projectile System Walkthrough

This section details the calculation steps using RK4, from initial velocity setup to updating the state with a weighted average of the slopes (k values), and finally providing a summary of one timestep's calculation.

17 Physics Mechanics Demonstrations

17.1 Force Application to Velocity

Demonstrates the conversion of applied force into initial speed and direction, affecting the projectile's trajectory.

–Set the initial conditions for projectile motion by adjusting the launch angle (θ) and the force (F), determining the initial velocity ($v_0 = \frac{F}{m}$), where m is the projectile's mass.

17.2 Elastic Collision and Momentum Transfer

Details the simulation of elastic collisions between entities, showcasing momentum conservation and transfer according to physics laws.
–Elastic collisions conserve both momentum and kinetic energy. For two colliding objects, momentum conservation is expressed as $m_1 v_{1,initial} + m_2 v_{2,initial} = m_1 v_{1,final} + m_2 v_{2,final}$, and kinetic energy conservation as $\frac{1}{2} m_1 v_{1,initial}^2 + \frac{1}{2} m_2 v_{2,initial}^2 = \frac{1}{2} m_1 v_{1,final}^2 + \frac{1}{2} m_2 v_{2,final}^2$.

18 Challenges and Improvements

18.1 Challenges Faced

- Implementing the RK4 method for trajectory calculation.
- Developing a functional stepper function for physics updates.
- Accurate momentum transfer and collision detection among entities.

18.2 Proposed Improvements

- Introduce more sophisticated computation logic, possibly utilizing piecewise functions or parametric equations to enrich the gameplay mechanics.
- Expand the physics simulations to include additional forces like wind and drag.
- Enhance visuals and user experience by adding diverse levels, implementing time-limits for launches, and refining the graphical interface.

19 Pseudo Code for Physics Simulation

Listing 1: Physics System Simulation

```
#include <boost/numeric/odeint.hpp>
#include <array>
// Define the state type for the simulation
using State = std::array<double, 4>; // x, y, vx, vy
using Stepper = boost::numeric::odeint::runge_kutta4<State>;

void projectileSystem(const State &state, State &dstate_dt, const double) {
    dstate_dt[0] = state[2]; // dx/dt = vx
    dstate_dt[1] = state[3]; // dy/dt = vy
    dstate_dt[2] = 0;        // dvx/dt = 0
    dstate_dt[3] = GRAVITY;  // dvy/dt = gravity
}
```

20 Detailed Physical System Description

Building on PS1 and PS2 from the Physical System Description, let's elaborate:

- PS1: The simulation engine represents a physical environment where forces such as gravity, tension, and friction are simulated in real time to affect objects within the game.
- PS2: The user interface includes a viewport of the simulation, input controls for adjusting force and angle, and feedback displays for velocity and trajectory.

21 Enhanced Solution Characteristics Specification

21.0.1 Assumptions

A3: The system shall have access to sufficient computational resources to perform real-time simulations.

A4: The user's input device is accurate and allows for precise control over input parameters.

21.0.2 Enhanced Data Definitions

In addition to DD1 and DD2, we consider:

DD3: Environmental Data - Includes parameters like wind speed and direction that affect the trajectory of projectiles.

21.0.3 Enhanced Instance Models

Building on IM1 and IM2:

IM3: Wind Influence Model - Integrates environmental data to adjust the projectile's trajectory accordingly.

22 Enhanced Requirements

22.1 Additional Functional Requirements

5. The system shall simulate the impact of environmental factors like wind on projectile trajectories.
6. The system shall provide a tutorial mode that explains the physics concepts through interactive examples.

22.2 Additional Non-functional Requirements

4. **Scalability:** The game shall be designed to easily integrate new physics models and game levels.

23 Additional System Features

23.1 Environmental Effects Simulation

1. The game will simulate environmental effects, providing players with challenges such as compensating for wind when aiming projectiles.

24 User Documentation and Help System

1. Comprehensive user documentation will be provided, explaining how to play the game and understand the underlying physics concepts.
2. An in-game help system will offer hints and explanations for each level, aiding users in comprehending the application of physics in gameplay.

25 Development Plan

The development of "Angry Birds Alike" will follow these phases:

1. Requirements analysis and project setup.
2. Core physics engine and gameplay mechanics development.
3. Implementation of user interface and graphics.
4. In-depth testing across platforms, fine-tuning based on feedback.
5. Final preparations for launch, including marketing and creation of support documentation.
6. Post-launch support and development of additional content based on user feedback.

26 Traceability Information and Coverage

This matrix links SRS requirements, MG modules, and VnV test cases to ensure comprehensive coverage and coherence across documentation.

SRS Requirement	**MG Module**	**VnV Test Case**
IM1: Projectile Motion	M4: Physics Engine	TC1: Projectile Motion Accuracy
IM2: Collision Dynamics	M4: Physics Engine	TC2: Collision Dynamics
IM3: Gravity Effects	M4: Physics Engine	TC1: Gravity Effects Verification

Table 7: Traceability Matrix linking SRS Requirements, MG Modules, and VnV Test Cases.

27 Appendices

27.1 Appendix A: Glossary

SRS: Software Requirements Specification.

RK4: Runge-Kutta 4th Order Method, a numerical technique used for solving ordinary differential equations.

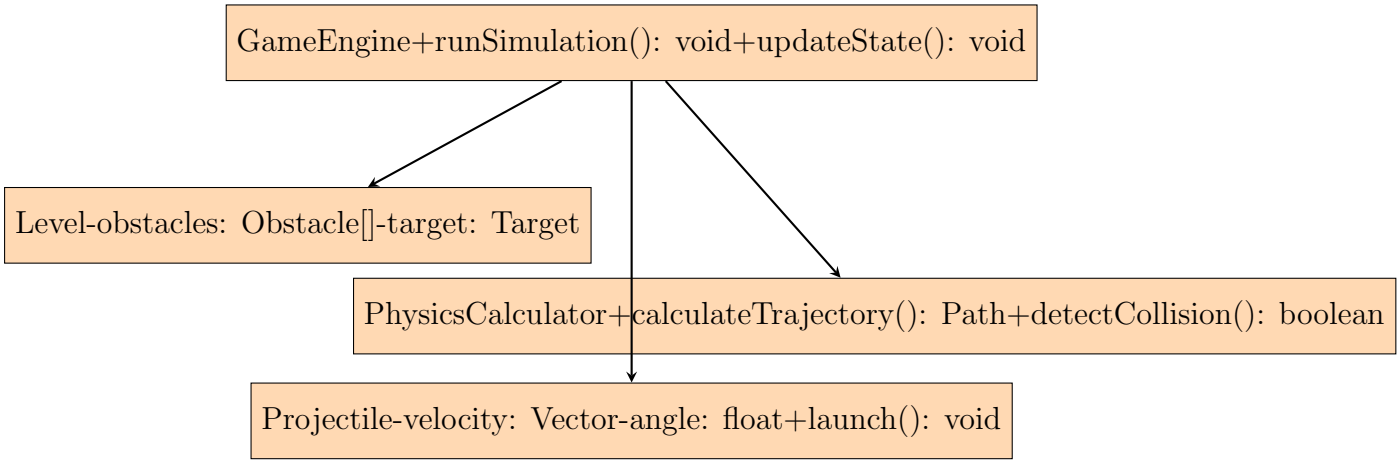
SFML: Simple and Fast Multimedia Library, a cross-platform software development library designed for creating multimedia applications.

ODE: Ordinary Differential Equation, a differential equation containing one or more functions of one independent variable and its derivatives.

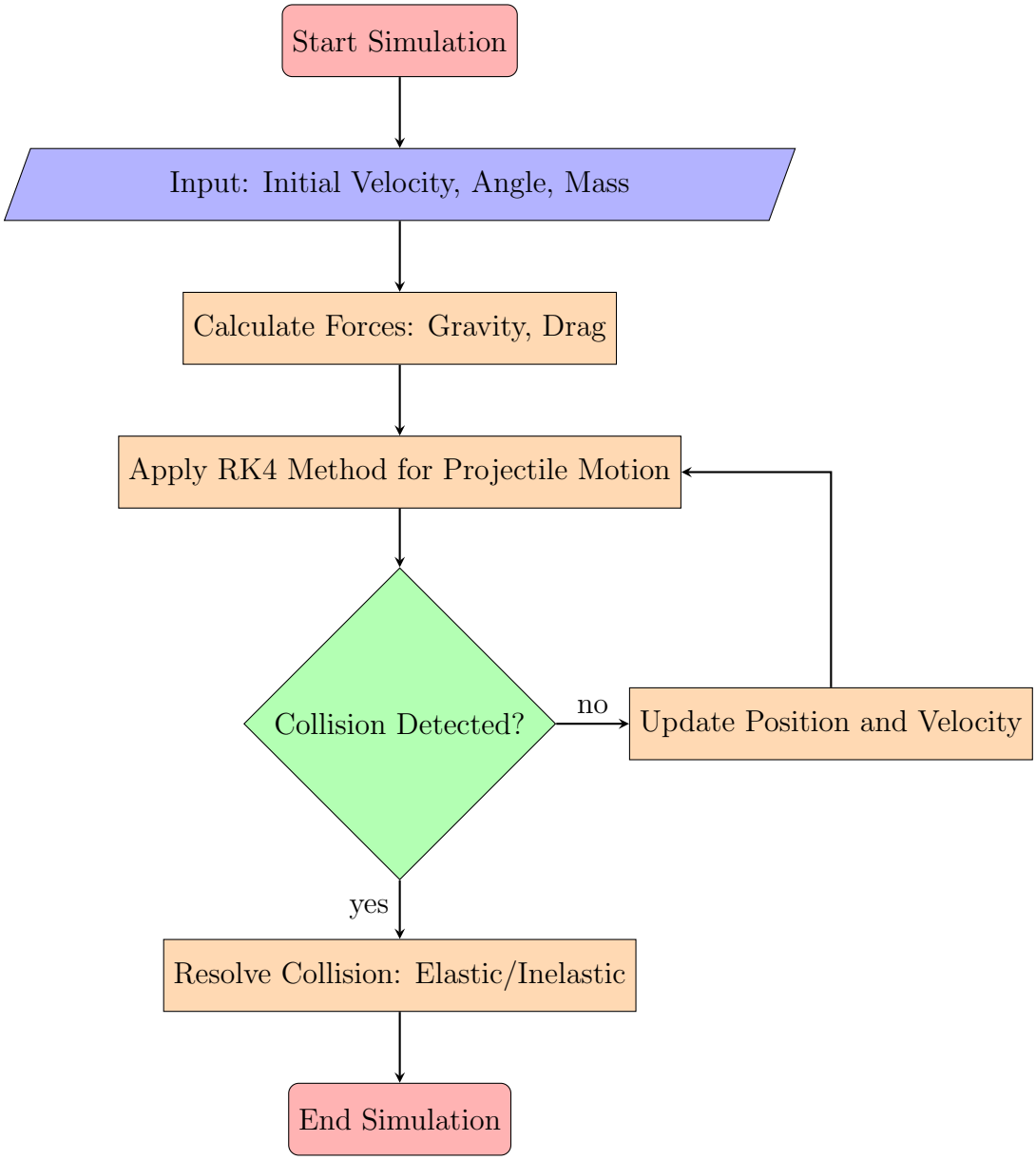
27.2 Appendix B: Analysis Models

27.2.1 DataFlow Diagram

27.2.2 Class Diagram



27.2.3 Physics Calculation Flowchart



27.3 Appendix C: Issues List

Issue ID	Description	Resolution
001	some inaccuracy found as projectile trajectory predictions at edge cases when used with Euler	Implemented improved numerical integration technique Like RK4 for edge cases.

27.4 Appendix D: Compliance Requirements

- The game must comply with the General Data Protection Regulation (GDPR) for users in the European Union.

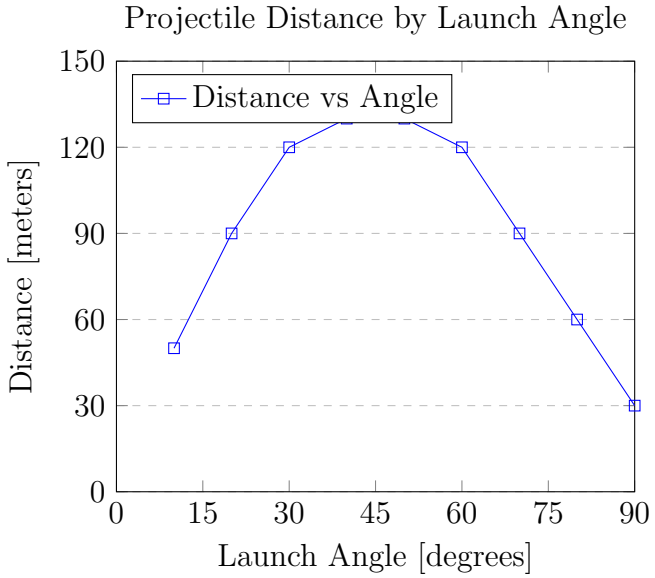
- Accessibility features must align with the Web Content Accessibility Guidelines (WCAG) 2.1 for inclusivity.

27.5 Appendix E: Detailed RK4 Method Explanation

The Runge-Kutta 4th Order Method is used to approximate the solutions to the ordinary differential equations governing projectile motion in "Angry Birds Alike". Here is a step-by-step breakdown of the RK4 method:

27.6 Projectile Distance by Launch Angle

The following chart illustrates the distance traveled by a projectile over a range of launch angles at a fixed initial speed, demonstrating the optimal angle for maximum distance.



28 References

The following sources were referenced in the preparation of this Software Requirements Specification:

1. SFML Documentation. (n.d.). *Simple and Fast Multimedia Library*. Retrieved from `https://www.sfml-dev.org/documentation/`

2. Boost.odeint Documentation. (n.d.). *odeint - Solving Ordinary Differential Equations*. Retrieved from `https://www.boost.org/doc/libs/1_75_0/libs/numeric/odeint/doc/html/index.html`

3. Hairer, E., Nørsett, S. P., & Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics.

4. Rovio Entertainment Corporation. (n.d.). *Angry Birds*. Retrieved from `https://www.angrybirds.com/`

5. Serway, R. A., & Jewett, J. W. (2018). *Physics for Scientists and Engineers.* Cengage Learning.

6. Real-Time Collision Detection. (2005). Christer Ericson, *Real-Time Collision Detection.* Morgan Kaufmann Publishers.

7. Butcher, J. C. (2008). *Numerical Methods for Ordinary Differential Equations.* John Wiley & Sons.

8. The Physics Classroom. (n.d.). *Momentum and Its Conservation.* Retrieved from `https://www.physicsclassroom.com/class/momentum`

9. Kiusalaas, J. (2013). *Numerical Methods in Engineering with Python 3.* Cambridge University Press.

10. OpenGL Wiki. (n.d.). *Main Page.* Retrieved from `https://www.khronos.org/opengl/wiki/Main_Page`