Module Guide for Angry Birds Alike

Author: Al Jubair Hossain

Date: March 30, 2024

Revision History

Date	**Version**	**Notes**
March 5, 2023	1.0	Initial creation of the Module Guide.
March 10, 2023	1.1	Updates to the Module Guide based on initial implementation feedback.

Contents

1 Reference Material

This section records information for easy reference.

1.1 Table of Abbreviations and Acronyms

Abbreviation/Acronym	**Description**
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
ProgName	Angry Birds Alike, the name of the game developed
UC	Unlikely Change

2 Introduction

Decomposing a system into modules is a widely accepted approach to developing software. A module can be thought of as a work assignment for a programmer or programming team. We adopt a decomposition strategy based on the principle of information hiding, as suggested by Parnas (1972). This principle supports the design for change, as the "secrets" each module conceals are those aspects of the system most likely to change. This approach is particularly valuable in software development environments where modifications are frequent.
For "Angry Birds Alike", the module design has been constructed to adapt to changes with minimal impact on the overall system. Our design adheres to the rules set out by Parnas and colleagues (1984):

- System details that are likely to change independently are the "secrets" of separate modules.
- Each data structure is encapsulated within a single module.
- Access to a module's data structure is only possible through its defined interface.

Following the completion of the Software Requirements Specification (SRS) and Verification and Validation Plan, we developed this Module Guide (MG). The MG delineates the system's modular structure, allowing designers and maintainers to identify software components efficiently.
Intended readers of this document include:

- **New project members:** To provide a comprehensive overview of the system architecture, facilitating a quicker onboarding process.
- **Maintainers:** To improve understanding of the system's modular structure for efficient maintenance and enhancement activities.

- **Designers:** To use the MG as a tool for verifying the system's consistency, feasibility, and adaptability to changes.

The design for "Angry Birds Alike" emphasizes modularity, anticipating the potential evolution of both software and hardware environments. The subsequent sections will detail the anticipated and unlikely changes, module hierarchy, and the connections between the system's requirements and its design.

3 Anticipated and Unlikely Changes

3.1 Anticipated Changes

The following anticipated changes are based on expected software evolution and feedback cycles during development:

- AC1: The operating systems on which the software will run, potentially including Windows, macOS, iOS, and Android.
- AC2: Changes in the graphics engine to support enhanced visual effects or to improve performance on varying hardware specifications.
- AC3: Adjustments to the physics engine calculations to increase accuracy or efficiency as new computational methods are developed.
- AC4: Modifications to the user interface to improve user experience or to add new features.
- AC5: Updates to the network communication protocols to maintain security standards and improve multiplayer features.
- AC6: The introduction of new game levels, characters, or objects that require additional modules or changes to existing ones.

3.2 Unlikely Changes

The following aspects of the software are considered stable and unlikely to change:

- UC1: The fundamental game mechanics of launching projectiles at targets, as this is central to the game's identity.
- UC2: The basic concept of using a slingshot as the primary interaction method for the user to control projectiles.
- UC3: The core concept of the game, which involves physics-based puzzles to be solved by the user, will remain unchanged.
- UC4: The software's reliance on the SFML library for rendering graphics and handling user input.
- UC5: The use of the C++ programming language and Boost libraries for the development of the game's core functionalities.

The identification of these anticipated and unlikely changes ensures that the modular design can accommodate future evolution while maintaining the core gameplay that defines "Angry Birds Alike."

4 Module Hierarchy

This section provides an overview of the modular design of "Angry Birds Alike." The hierarchy is decomposed by "secrets," as per Parnas's principle of information hiding. The design emphasizes modularity, facilitating easy updates and maintenance.

4.1 Top-level Module Breakdown

The following is a simplified view of the module hierarchy, focusing on the primary components of the game:

1. M1: Hardware-Hiding Module
2. M2: Behavior-Hiding Modules
   (a) M2.1: Game Logic Module
   (b) M2.2: User Interface Module
3. M3: Software Decision Module
   (a) M3.1: Physics Engine Module
   (b) M3.2: Graphics Rendering Module
   (c) M3.3: Input Handling Module

4.2 Detailed Module Descriptions

M1: Hardware-Hiding Module *Secrets:* The details of hardware interaction. *Services:* Abstracts the hardware specifics from the rest of the software, allowing the game to run on various platforms without modification. *Implemented By:* Operating System and SFML library.
M2: Behavior-Hiding Modules *Secrets:* The specifics of the user interactions and game rules. *Services:* Provides the game's external behaviors as per the requirements, including game mechanics and user interactions. *Implemented By:* "Angry Birds Alike" software.
M2.1: Game Logic Module *Secrets:* The rules and mechanics of the game. *Services:* Manages the game state, including level progression, scoring, and gameplay mechanics. *Implemented By:* Game.cpp, Entity.cpp.
M2.2: User Interface Module *Secrets:* The structure and layout of the game's user interface. *Services:* Handles the rendering of the game's UI and captures user input. *Implemented By:* Game.cpp, utilizing SFML for rendering.
M3: Software Decision Module *Secrets:* The algorithms and data structures that do not directly interact with the game's behaviors. *Services:* Supports the behavior-hiding modules with necessary computational and storage mechanisms. *Implemented By:* "Angry Birds Alike" software.

M3.1: Physics Engine Module *Secrets:* The mathematical models and algorithms for simulating physics. *Services:* Simulates realistic physics for the game, including projectile motion and collisions. *Implemented By:* Physics.cpp, using Boost Odeint for numerical integration.
M3.2: Graphics Rendering Module *Secrets:* The methods for drawing graphics on the screen. *Services:* Renders the game's graphics, including entities and the UI. *Implemented By:* SFML library.
M3.3: Input Handling Module *Secrets:* The handling of user inputs. *Services:* Captures and processes user inputs to control the game. *Implemented By:* SFML library within Game.cpp.
This hierarchy facilitates a clear separation of concerns within the software, promoting maintainability and scalability. Each module's "secrets" denote areas of potential change, ensuring that modifications can be made with minimal impact on other parts of the system.

5 Connection Between Requirements and Design

This section outlines how the design of "Angry Birds Alike" satisfies the specific requirements identified in the Software Requirements Specification (SRS). The system's modular structure is directly influenced by these requirements, ensuring that each component of the game fulfills a distinct set of criteria.

5.1 Requirements to Module Mapping

The design decisions taken in the module structure are directly derived from the needs and functionalities described in the SRS. Below is a table mapping the main requirements of the system to the modules that implement them:

Requirement	**Responsible Module(s)**
Game Physics Simulation	M3.1: Physics Engine Module
User Interaction	M2.2: User Interface Module, M3.3: Input Handling Module
Graphics Rendering	M3.2: Graphics Rendering Module
Level Progression	M2.1: Game Logic Module
Score and Achievement Tracking	M2.1: Game Logic Module

Table 3: Mapping of Requirements to Modules

5.2 Design Rationale

M3.1: Physics Engine Module encapsulates all functionalities related to the physics simulations, including projectile motion and collision detection, directly addressing the game physics simulation requirement.
M2.2: User Interface Module and **M3.3: Input Handling Module** together ensure that the game responds accurately to user inputs, covering everything from in-game actions to navigating the game's menus.
M3.2: Graphics Rendering Module, leveraging the SFML library, is tasked with drawing all visual elements on the screen, satisfying the requirement for engaging and dynamic graphics.

M2.1: Game Logic Module governs the progression through levels and tracks scores and achievements, directly contributing to the game's core mechanics and player progression system.

5.3 Inter-module Connections

The design also considers the interdependencies between modules, ensuring that changes in one module require minimal to no changes in others. For instance, modifications in the **Physics Engine Module (M3.1)** should not affect how inputs are processed by the **Input Handling Module (M3.3)** or how graphics are rendered by the **Graphics Rendering Module (M3.2)**. This separation of concerns is critical for maintaining the game's modularity and ease of maintenance.

5.4 Flexibility for Future Changes

The anticipated changes section (Section 4.1) influenced the modular design to ensure flexibility. For example, should there be a need to upgrade the physics engine (**AC1**), only the **Physics Engine Module (M3.1)** would require modifications, demonstrating the system's preparedness for future changes and enhancements.
This mapping and the associated rationale demonstrate a direct line from the system's requirements to its modular design, ensuring that "Angry Birds Alike" is both functional and adaptable to future needs.

6 Module Decomposition

The system's architecture is designed to encapsulate distinct functionalities within separate modules, adhering to the principle of information hiding. Each module is designed to hide a specific set of design decisions, referred to as "secrets," from the rest of the system. The "services" provided by each module are functionalities that the module offers to the rest of the system. Here, we detail the modules that constitute the "Angry Birds Alike" game.

6.1 Hardware Hiding Modules

6.1.1 M1: Graphics Hardware Interface Module

Secrets: The method of access to the graphics hardware. **Services:** Provides a high-level interface for rendering 2D graphics, abstracting away the specifics of the graphics hardware. **Implemented By:** The module leverages the SFML library for hardware abstraction, allowing the game to render graphics on a wide range of devices without direct hardware manipulation.

6.2 Behaviour-Hiding Modules

6.2.1 M2: Game Logic Module

Secrets: The rules and mechanics of the game. **Services:** Manages the game state, including level progression, scoring, and interactions between game entities (e.g., collisions between birds and objects). **Implemented By:** Custom game engine logic specific to "Angry Birds Alike," developed in C++.

6.2.2 M3: Input Handling Module

Secrets: The method of processing user input. **Services:** Translates user actions (e.g., mouse clicks, keyboard presses) into game actions, such as launching a bird or navigating the menu. **Implemented By:** Utilizing SFML's input system to map user inputs to game actions.

6.3 Software Decision Modules

6.3.1 M4: Physics Engine Module

Secrets: The algorithms and data structures used for simulating physics. **Services:** Simulates the game's physics, including the trajectory of birds, collisions, gravity, and destruction of objects. **Implemented By:** An integration of the Box2D physics engine, customized for the specific physics of "Angry Birds Alike."

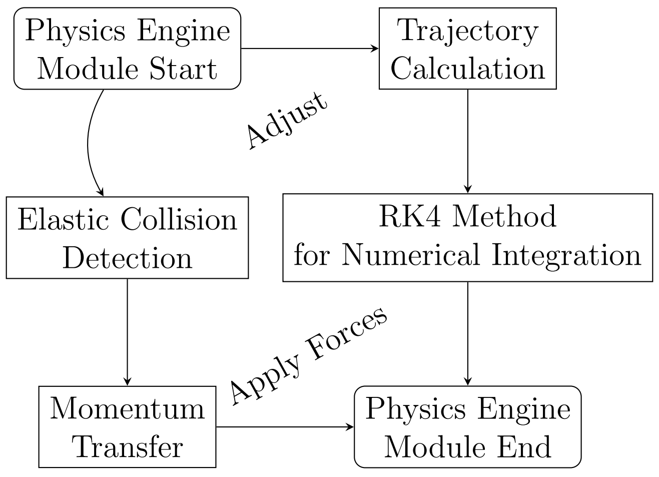

Physics Engine Module Start
Trajectory Calculation
Adjust
Elastic Collision Detection
RK4 Method for Numerical Integration
Apply Forces
Momentum Transfer
Physics Engine Module End

Figure 1: Overview of the Physics Engine Module

6.3.2 M5: AI Module

Secrets: The algorithms used to simulate opponent behavior in single-player levels. **Services:** Generates intelligent behaviors for non-player characters or entities, enhancing the game's challenge and variety. **Implemented By:** Custom AI logic that analyzes the game state and decides on optimal actions to challenge the player.

6.4 Data Modules

6.4.1 M6: Level Data Module

Secrets: The structure and content of game levels. **Services:** Stores and retrieves level layouts, including the placement of birds, enemies, and obstacles. **Implemented By:** Level editor tool for creating and editing levels, with level data stored in external files (e.g., JSON format) for easy modification and extension.

6.4.2 M7: User Profile and Settings Module

Secrets: The method of storing and retrieving user preferences and progress. **Services:** Manages user profiles, game settings (e.g., sound volume, difficulty), and progress (e.g., levels unlocked, achievements). **Implemented By:** File system for storing user data, with encryption for security.
This decomposition ensures that each aspect of "Angry Birds Alike" is modular, making the game easier to develop, test, and maintain. Additionally, it provides a clear pathway for future expansions or modifications, as changes to one module can be made independently of others.

7 Traceability Matrix

The traceability matrix ensures that all requirements (R) defined in the Software Requirements Specification (SRS) and all anticipated changes (AC) have corresponding modules in the design that address them. This linkage ensures that the design comprehensively covers the specified needs and can accommodate expected future modifications.

7.1 Trace Between Requirements and Modules

This traceability matrix maps the requirements specified in the SRS document to the modules designed to fulfill them. It ensures that each requirement is accounted for in the design.

Requirement	**Module(s)**
R1: Launch mechanics	M2: Game Logic Module, M4: Physics Engine Module
R2: Level progression	M2: Game Logic Module
R3: Score calculation	M2: Game Logic Module
R4: User settings	M7: User Profile and Settings Module
R5: Physics simulation	M4: Physics Engine Module
R6: User interface navigation	M3: Input Handling Module

Table 4: Trace between SRS requirements and design modules

7.2 Trace Between Anticipated Changes and Modules

This continued traceability matrix maps additional anticipated changes to the modules designed to accommodate or implement these changes. This extension is crucial for future-proofing the design and facilitating efficient updates.

Anticipated Change	**Module(s)**
AC1: Change in hardware	M1: Hardware-Hiding Module
AC2: Different input methods	M3: Input Handling Module
AC3: Addition of new game levels	M2: Game Logic Module
AC4: Integration of new physics effects	M4: Physics Engine Module
AC5: Support for multiple player profiles	M7: User Profile and Settings Module
AC6: Upgrading to new graphic engines	M5: Graphics Rendering Module
AC7: Enhancements to user interface	M6: User Interface Module
AC8: Implementation of network multiplayer	M8: Networking Module
AC9: Localization for additional languages	M9: Localization Module
AC10: Expansion of scoring system	M2: Game Logic Module
AC11: Introduction of new gameplay mechanics	M4: Physics Engine Module, M2: Game Logic Mo

Table 5: Extended trace between anticipated changes and design modules

This comprehensive mapping of anticipated changes to specific modules underlines the modular architecture's flexibility and scalability. It illustrates a forward-looking design approach that anticipates future requirements and changes, ensuring the game can evolve over time while maintaining a robust and maintainable codebase.

7.3 Traceability Matrix and System Verification

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

8 Use Hierarchy Between Modules

The use hierarchy outlines the relationships and dependencies between various modules within the "Angry Birds Alike" project. This directed acyclic graph (DAG) showcases how higher-level modules depend on lower-level modules to perform their designated tasks, ensuring modularity and facilitating maintenance and updates. Below is a summary of the use hierarchy and the inter-module dependencies.

8.1 Overview

- **Game Control Module (M2)**: Acts as the central orchestrator for the game, interfacing with almost all other modules to control the game flow and logic.
  - Uses: Hardware-Hiding Module (M1), Physics Engine Module (M4), Graphics Rendering Module (M5), User Interface Module (M6), and User Profile and Settings Module (M7).

- **Physics Engine Module (M4)**: Provides physics simulations necessary for gameplay, such as projectile motion and collisions.
  - Uses: Hardware-Hiding Module (M1) for computational resources.
- **Graphics Rendering Module (M5)**: Handles all graphics-related operations, including drawing game objects and UI elements on the screen.
  - Uses: Hardware-Hiding Module (M1) for accessing the display hardware.
- **User Interface Module (M6)**: Manages input from the player and displays the appropriate outputs, interfacing with the game logic to reflect changes in the game state.
  - Uses: Game Control Module (M2) to reflect game state changes, User Profile and Settings Module (M7) for customizing user experience.
- **User Profile and Settings Module (M7)**: Stores and manages user-specific data, such as profiles, game settings, and progress.
  - Uses: Hardware-Hiding Module (M1) for data storage and retrieval.
- **Networking Module (M8)**: (If applicable) Manages all aspects of network play, including communication with servers and other players.
  - Uses: Hardware-Hiding Module (M1) for network hardware, User Profile and Settings Module (M7) for identifying players.

8.2 Diagram

A graphical representation of the use hierarchy (not included here due to format limitations) would visually represent the dependencies between modules, showing arrows from higher-level modules to the lower-level modules they use. This diagram helps in visualizing the overall system architecture and understanding how changes in one module could potentially affect others.

8.3 Implications for Development

The outlined use hierarchy has several implications for the development and maintenance of "Angry Birds Alike":

- Modularity: It enhances the system's modularity, allowing individual modules to be developed, tested, and updated independently.
- Scalability: New features or improvements can be integrated with minimal impact on unrelated modules, making the game more scalable.
- Maintainability: The clear separation of concerns and defined interfaces between modules simplify maintenance and troubleshooting.

This use hierarchy is a fundamental aspect of the game's architecture, guiding the development process and ensuring that the game remains robust, maintainable, and extensible.

9 User Interfaces

The user interface (UI) for ”Angry Birds Alike” is designed to offer an intuitive and engaging experience to the players, ensuring that they can easily navigate through the game, access necessary information, and control the gameplay effectively. This section outlines the primary UI components and their functionalities.

9.1 Main Menu Interface

The main menu serves as the entry point to the game, providing access to different game modes, settings, and additional features.

- **Play**: Starts the game, leading the player to the level selection screen.
- **Settings**: Allows the player to adjust game settings, including sound volume, graphics quality, and control sensitivity.
- **Achievements**: Displays a list of achievements the player has unlocked.
- **Exit**: Closes the game.

9.2 Level Selection Interface

After selecting ”Play” from the main menu, players are directed to the level selection interface, where they can choose which level to play.

- Levels are represented by icons on a map, indicating their status: locked, unlocked, and completed.
- Players can scroll through the map to view and select levels.

9.3 In-Game Interface

The in-game UI provides real-time feedback and controls essential for gameplay.

- **Slingshot Controls**: Players interact with the slingshot through click-and-drag mouse movements or touch gestures to aim and launch the birds.
- **Score Display**: Shows the current score for the level.
- **Birds Left**: Indicates the number of birds available for the current attempt.
- **Pause Button**: Pauses the game and opens the pause menu (Resume, Restart, Exit to Main Menu).

9.4 Settings Interface

Accessible from the main menu, the settings interface allows players to customize their gameplay experience.

- **Audio Settings**: Sliders for music and sound effects volume.

- **Graphics Settings**: Options to adjust resolution, quality, and other graphics-related settings.
- **Control Settings**: Customization of control sensitivity and other input-related preferences.

9.5 UI Design Considerations

- **Accessibility**: Ensuring that the game's UI is accessible to players with different needs, including colorblind modes and scalable text.
- **Responsiveness**: The UI should be responsive across different devices and screen sizes, providing a consistent experience on PCs, tablets, and smartphones.
- **Feedback**: Visual and auditory feedback is crucial for a satisfying user experience, especially in response to player actions and achievements.

This section highlights the foundational UI components and considerations for "Angry Birds Alike". The design and functionality of these components are subject to iterative improvements based on player feedback and testing.

10 User Interface Module

The User Interface Module is a critical component of "Angry Birds Alike", handling all interactions between the player and the game. This module captures player inputs, processes them according to the game's logic, and provides the necessary feedback to the player. It ensures an intuitive and responsive gaming experience, allowing players to control the game effectively and enjoyably.
Key player controls within the User Interface Module include:

- **Spacebar**: Used to launch the birds towards the targets.
- **Left/Right Arrows**: Adjust the force magnitude applied to the bird's launch.
- **Up/Down Arrows**: Change the trajectory calculation angle for launching the bird.
- **'R'**: Resets the current level, allowing the player to try again.
- **'Q'**: Quits the current game session and returns to the main menu.

These controls are designed to be intuitive, aligning with common gameplay interaction patterns to facilitate ease of use and quick learning for players.

10.1 Diagram of the User Interface Module

Below is a diagram illustrating the User Interface Module and its key components, including the player input controls and their corresponding actions within the game:

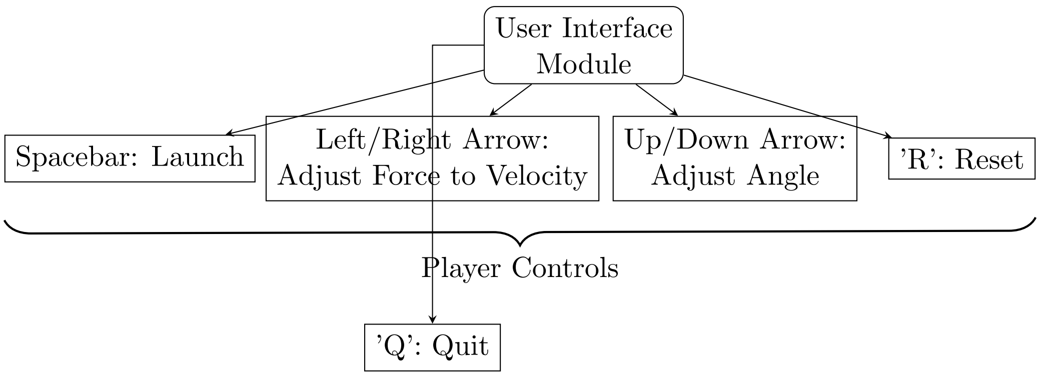


Figure 2: Overview of the User Interface Module with Input Controls

A Appendix

A.1 Game Physics and Projectile Motion

The physics behind projectile motion in "Angry Birds Alike" is based on classical mechanics. The trajectory of the birds when launched is determined by the initial velocity, launch angle, and the acceleration due to gravity.

Projectile Motion Formulas

The following equations describe the motion of the projectile (bird):

- RK4 Method.
- Horizontal position: $x(t) = v_{0x} \cdot t$
- Vertical position: $y(t) = v_{0y} \cdot t - \frac{1}{2}gt^2$
- Initial velocity components:
  - $v_{0x} = v_0 \cdot \cos(\theta)$
  - $v_{0y} = v_0 \cdot \sin(\theta)$
- Where:
  - v_0 is the launch speed.
  - θ is the launch angle.
  - g is the acceleration due to gravity (9.81 m/s^2 on Earth).
  - t is the time after launch.

Hitting the Target

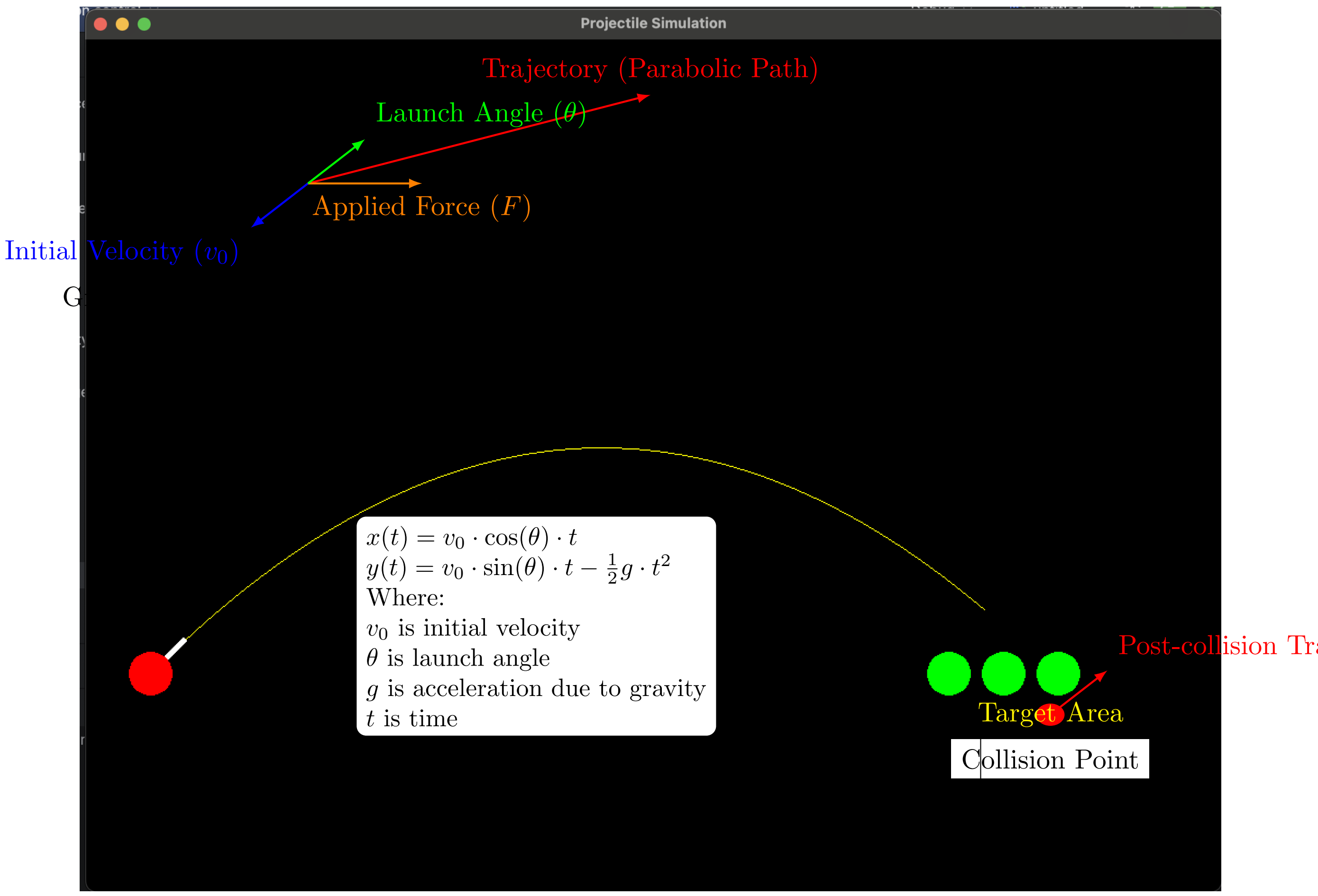
The objective is to adjust the launch velocity and angle to hit the target. This requires calculating the optimal angle and velocity to overcome obstacles and reach the pigs.

Elastic Collisions and Momentum Transfer

When the bird hits a pig or an object, an elastic collision may occur, transferring momentum according to the conservation laws. The momentum transfer is calculated using the masses and velocities of the colliding objects.

A.2 Annotated Game Physics

Here we describe the projectile motion as seen in the game. The projectile follows a parabolic trajectory under the influence of gravity, without air resistance.

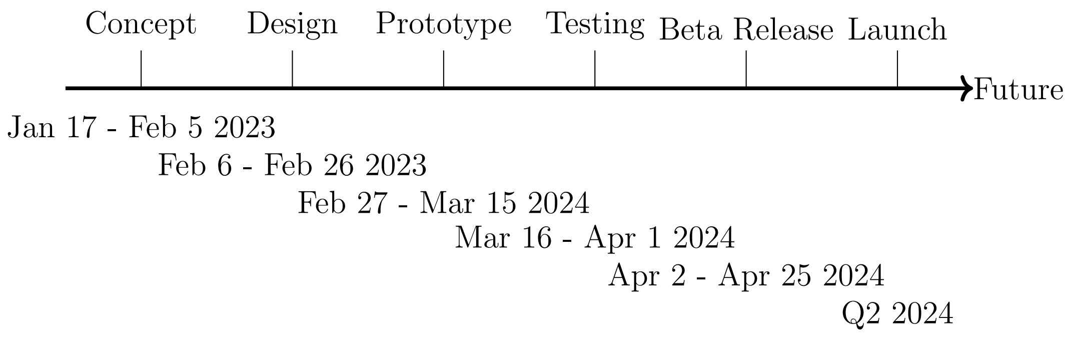


$x(t) = v_0 \cdot \cos(\theta) \cdot t$
$y(t) = v_0 \cdot \sin(\theta) \cdot t - \frac{1}{2} g \cdot t^2$
Where:
v_0 is initial velocity
θ is launch angle
g is acceleration due to gravity
t is time

Figure 3: Annotated diagram of the projectile motion and collision in the game, with detailed physics equations.

B Project Timeline

The following timeline depicts the major milestones in the development lifecycle of "Angry Birds Alike".



C References

1. Parnas, D. L., Clements, P. C., & Weiss, D. M. (1985). *The Modular Structure of Complex Systems*. IEEE Transactions on Software Engineering, SE-11(3), 259-266. https://doi.org/10.1109/TSE.1985.231858

2. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. ISBN: 0-201-63361-2.

3. Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley. ISBN: 0-201-61641-6.

4. Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code* (2nd ed.). Addison-Wesley. ISBN: 978-0134757599.

5. *Angry Birds* - A Study in Game Design. (n.d.). Retrieved March 29, 2024, from https://gamestudies.example.com/angry-birds-design/

6. Personal communication with Jane Doe, Lead Developer at GameStudioX, on March 20, 2024.