Module Guide for Physics Game Based on Angry Birds

Al Jubair Hossain

March 15, 2024

1 Revision History

Date	Version	Notes
March 4, 2024	1	Changed section (Hyperlink)
March 12, 2024	2	Changed section (7)

Contents

List of Tables

List of Figures

2 Introduction

2.1 Purpose of the Module Guide

The purpose of this Module Guide is to provide an in-depth understanding of the software modules that constitute the Sample Physics-Based Game. It outlines the individual modules' functionalities, dependencies, interfaces, and relationships, offering insights into the game's architecture and design. The guide is intended for developers, architects, and other stakeholders involved in the development, testing, and maintenance of the game.

2.2 Scope

This Module Guide focuses on documenting the specifics of each software module within the Sample Physics-Based Game. It covers modules related to physics simulation, game elements, user interaction, graphics, and sound. The scope includes detailed descriptions of each module's purpose, functionalities, and interactions with other modules.

3 Game Overview

3.1 Game Description

The game is a physics-based gaming application designed to provide players with an engaging and challenging experience. Players interact with the game environment by manipulating objects, analyzing trajectories, and applying physics principles. The central focus is on realistic collision dynamics, gravitational forces, and other physical phenomena..

3.2 Physics Simulation

The heart of the game lies in its physics simulation, where realistic collision interactions between a projectile and targets/obstacles are implemented. Gravity influences the trajectory and motion of the projectile, adding an authentic touch to in-game dynamics. Additionally, the game introduces the option to consider additional forces, such as wind, for heightened complexity. Visual feedback for collisions, including deformation or destruction of targets, enhances the overall gaming experience.

3.3 Game Architecture

3.3.1 High-Level Components

- **User Input:** Receives and interprets input from players, allowing them to interact with the game.
- **Launch Controls:** Manages the parameters for launching the projectile, providing options for adjusting launch angle, force, and other relevant parameters.

- **Projectile Logic:** Governs the behavior and properties of the projectile, including its initial position, velocity, and mass.
- **Physics Engine:** Orchestrates the physics simulation, encompassing collision detection, gravity simulation, and the consideration of additional forces.
- **Graphics Module:** Enhances the visual appeal of the game, handling the rendering of objects and visual feedback for collisions.
- **Sound Effects:** Complements the gameplay experience by implementing auditory elements tied to in-game events.

3.3.2 Interactions between Components

- User Input influences Launch Controls, allowing players to control the launch parameters.
- Launch Controls communicate with Projectile Logic to define the projectile's characteristics.
- Physics Engine interacts with Projectile Logic for realistic collision interactions, gravity simulation, and handling additional forces.
- Graphics Module receives information from Physics Engine to render visual elements, providing feedback for collisions.
- Sound Effects are triggered based on events detected by the Physics Engine, enriching the auditory aspect of the game.

3.4 Dependencies on External Systems

The game is dependent on external systems, potentially involving compatibility with graphics libraries or leveraging external physics engines. Understanding and managing these dependencies are crucial for ensuring a seamless and optimized gaming experience.

4 Module Descriptions

4.1 Module

4.1.1 Purpose

The Bird module represents the player-controlled projectile in the game, defining its initial properties and behavior

4.1.2 Functionality

- Initialization of the bird's position, velocity, and mass.
- Handling user-triggered launches and applying realistic physics for projectile motion.
- Collision detection with game elements.

4.1.3 Dependencies

- Dependent on the game physics module for realistic collision interactions.
- Interfaces with the user interaction module to receive launch commands.

4.1.4 Interfaces

- Input Interface(s):
- User interaction module for launch commands.
- Output Interface(s):
- Interfaces with the game physics module for collision information.

4.1.5 Module Relationships

- Dependencies:
  - Relies on the game physics module for collision interactions.
- Dependents:
  - Other modules may depend on collision information from the Bird module.

5 Data Design

5.1 Physics Data Structures

5.1.1 Purpose

This section outlines the data structures employed for managing physics-related information within the game. The design focuses on representing projectile properties, collision data, and other parameters crucial for realistic physics simulation.

5.1.2 Data Structures

1. Projectile Properties:
    - Definition: Stores information such as position, velocity, and mass of the projectile.
    - Usage: Utilized by the Physics Engine to calculate and update the projectile's motion.
2. Collision Data:
    - Definition: Captures data related to collisions, including collision points and impact forces.
    - Usage: Enables accurate collision response and provides information for visual and auditory feedback.

5.2 Game State Management

5.2.1 Purpose

This section delves into the design of the game state management system, responsible for capturing and maintaining the overall state of the game. It encompasses information related to scores, levels, and the current status of game elements.

5.2.2 Data Structures

1. Game State
    - Definition: Represents the overall state of the game, including scores, level progression, and other relevant parameters.
    - Usage: Provides a centralized repository for managing and updating the current state of the game.

5.3 Input Data Handling

5.3.1 Purpose

Detailing the data structures and mechanisms employed for handling user input. This includes capturing and interpreting input from players to control game elements.

5.3.2 Data Structures

User Input Data:

- Definition: Stores information received from user inputs, such as mouse clicks or touch events.

- Usage: Feeds input data to the User Interface and Launch Controls for further processing.

5.4 Data Flow

5.4.1 Purpose

Explaining the flow of data within the game, illustrating how information moves between different components and modules

5.4.2 Data Flow Pathways

1. **User Input to Launch Controls:**
   - Flow: User Input data is transmitted to Launch Controls for interpreting player actions.
2. **Launch Controls to Projectile Logic:**
   - Flow: Launch parameters are communicated to Projectile Logic for configuring the projectile's initial properties.
3. **Physics Engine to Graphics Module:**
   - Flow: Information regarding collisions and physics simulations is conveyed to the Graphics Module for rendering visual feedback.
4. **Physics Engine to Sound Effects:**
   - Flow: Events detected by the Physics Engine trigger Sound Effects for auditory feedback.

6 Component Design

6.1 Component 1: Physics Simulation Component

6.1.1 Purpose

The Physics Simulation Component is designed to handle the realistic physics interactions within the game. It governs the motion of projectiles, detects collisions, and manages gravitational influences to create an authentic gaming experience.

6.1.2 Functionality

1. Projectile Motion:
    - Utilizes physics equations to calculate and update the motion of projectiles based on their initial conditions.
2. Collision Detection
    - Implements algorithms for detecting collisions between projectiles and targets/obstacles within the game environment.
3. Gravity Simulation:
    - Applies gravitational forces to influence the trajectory and motion of projectiles

6.1.3 Dependencies

- Physics Data Structures: Relies on data structures to access and manipulate physics-related information, including projectile properties and collision data.

6.1.4 Interfaces

1. **Input Interface:**
    - Receives input from the Launch Controls and Projectile Logic modules for initializing projectile properties.
2. **Output Interface:**
    - Communicates collision data to the Graphics Module for rendering visual feedback.

6.1.5 Interaction with Modules

- Launch Controls:
    - Receives launch parameters to set the initial conditions for projectiles.
- Projectile Logic:
    - Collaborates to integrate realistic physics calculations into the game's projectile behavior.

6.2 Component 2: User Input Component

6.2.1 Purpose

The User Input Component serves as the interface between players and the game, capturing and interpreting user inputs to control various aspects of gameplay.

6.2.2 Functionality

1. Input Capture:
    - Captures user inputs, including mouse clicks or touch events.
2. Input Interpretation:
    - Translates captured inputs into meaningful commands for the game.

6.2.3 Dependencies

None identified. The User Input Component operates independently of other components.

6.2.4 Interfaces

1. Output Interface:
    - Transmits interpreted user inputs to the Launch Controls for further processing.

6.2.5 Interaction with Modules

1. Launch Controls:
    - Provides interpreted user inputs to influence launch parameters and initiate projectile launches.

7 Quality Attributes

7.1 Physics Simulation Accuracy

7.1.1 Definition

Physics Simulation Accuracy evaluates the precision of in-game physics calculations, ensuring the faithful representation of real-world principles. It demands a meticulous implementation of equations such as the equations of motion $s = ut + \frac{1}{2}at^2$ for projectile motion and Newton's law of universal gravitation $F = Gm_1m_2/r^2$ for gravitational interactions.

7.1.2 Evaluation Criteria

- Accurate Projectile Motion Equations:
  - Implementing projectile motion equations with precision to model the realistic trajectory of objects in the game.
- Faithful Application of Newton's Law of Gravitation:
  - Ensuring that gravitational interactions closely follow Newton's law, contributing to the authenticity of in-game physics.

7.2 Gameplay Responsiveness

7.2.1 Definition

Gameplay Responsiveness measures the system's agility in responding to player inputs using principles like kinematics equations ($s = ut + \frac{1}{2}at^2$). It emphasizes minimizing the time between a player's action and the corresponding in-game reaction.

7.2.2 Evaluation Criteria

- Low Input-to-Action Latency with Kinematics:
  - Minimizing the time delay between a player's input (e.g., launching a projectile) and the resulting in-game action, applying kinematics equations.
- Consistent and High Frame Rate for Responsive Controls:
  - Maintaining a high and consistent frame rate to ensure fluid animations and responsiveness, drawing from principles of dynamics ($F = ma$).

7.3 Scalability

7.3.1 Definition

Scalability assesses the game's capability to handle increasing complexity and additional features, considering principles like force and acceleration ($F = ma$). It ensures optimal performance under varied scenarios.

7.3.2 Evaluation Criteria

- Optimal Performance under Increased Load using Newton's Second Law:
  - Ensuring the game performs optimally even with a larger number of projectiles, targets, or complex scenarios, applying Newton's second law of motion.

- Adaptability to New Features applying Force Equations:
  - Evaluating how easily new features or challenges can be integrated, incorporating force equations ($F = \Delta p/\Delta t$.)

7.4 Maintainability

7.4.1 Definition

Maintainability evaluates the ease of updating, enhancing, and debugging the game over its lifecycle. It emphasizes modular design and comprehensive documentation, akin to principles of work and energy (W = KE).

7.4.2 Evaluation Criteria

- **Modular Design for Work-Energy Principles:**
  - Utilizing modular design principles and organized code for ease of updates and modifications, inspired by principles of work and energy.
- **Comprehensive Documentation for Clear Understanding:**
  - Providing thorough documentation for clarity and understanding, reflecting the documentation practices in physics principles.

7.5 Usability

7.5.1 Definition

Usability assesses how easily players can interact with and enjoy the game. It considers intuitive controls and a clear user interface, applying principles of kinematics for control design and information visualization.

7.5.2 Evaluation Criteria

- Intuitive Controls with Kinematics Design:
  - Ensuring controls are easy to learn and intuitive, leveraging kinematics equations to design control mechanisms.
- Clear and Informative User Interface applying Information Visualization:
  - Providing a user interface that offers clear feedback and information, applying principles of information visualization for enhanced user understanding.

8 Physic equation Mockup

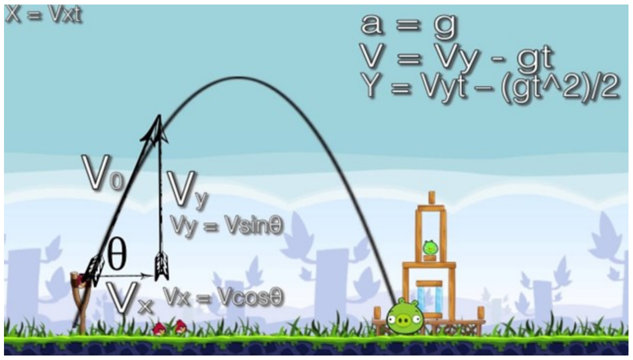

Figure 1: This image combines elements from 'Angry Birds' with physics equations to explain projectile motion, showing velocity components and trajectory formulas.

This image 2 appears to be a screenshot from a video playback of a game-based learning module or educational software. It displays a projectile motion scenario within a game setting. The vectors illustrate the physics concepts of initial velocity (v_i) and final velocity (v_f), with the dotted line indicating the trajectory of the launched object. The change in velocity $\delta(v)$ is represented by an arrow pointing upwards from the final to the initial velocity, demonstrating the change in the object's speed and direction as it moves. This educational tool seems designed to help users understand the principles of kinematics by visualizing the changes in an object's velocity during projectile motion. The high score and current score are displayed in the top right, implying a gamified learning experience.

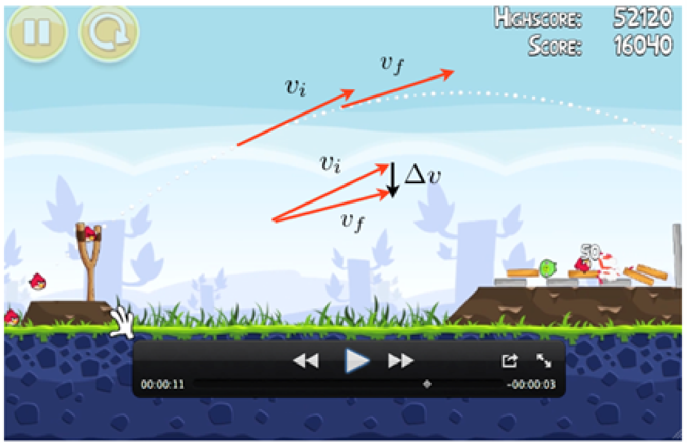

Figure 2: The image depicts a screenshot from 'Angry Birds' annotated with vectors to show initial velocity v_i, final velocity v_f, and the change in velocity $\Delta(V)$, demonstrating concepts of motion in physics.

The image 3 appears to be a playful educational overlay on a screenshot from a game, illustrating various principles of physics in the context of projectile motion. It includes equations and notations for energy (kinetic and potential), velocity (initial and final), acceleration due to gravity, angles, and trajectory paths. Notably:

$0 = \frac{1}{2}mv^2 + mg\Delta y$ - An energy conservation equation where kinetic energy plus potential energy equals zero, suggesting the highest point of the trajectory where velocity is zero.
$v = \sqrt{2gay}$ - An equation relating velocity to the acceleration due to gravity and the height of the projectile.

$a = 9.8$ m/s^2 - The acceleration due to Earth's gravity.
$v_x = \sqrt{(kx)^2/m} - 2gs\sin\theta$ - An equation for horizontal velocity taking into account spring constant, mass, gravity, and angle of launch.
$\tan\theta = v_y/v_x$ and $\theta = \tan^{-1}(v_y/v_x)$ - Equations for calculating the launch angle using the tangential function with vertical and horizontal velocity components.
$y_e = h + s\sin\theta$ - A trigonometric equation for the vertical displacement.

The image also includes a free body diagram for the angles of projection and force vectors at the launch point, and it visually depicts the projectile's parabolic path through the game's environment. The game's high score is visible, implying that understanding these physics concepts could be used to strategize and improve gameplay performance.

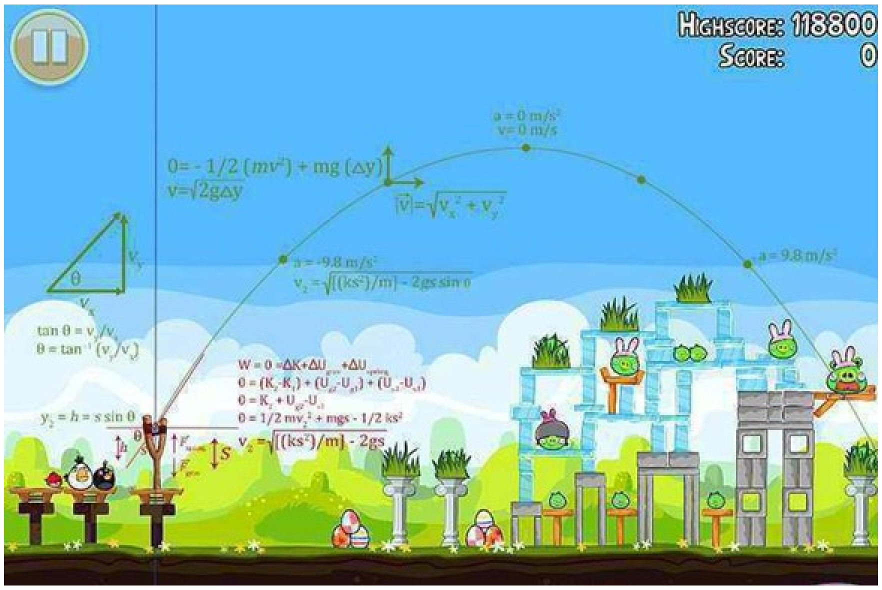

Figure 3: Use hierarchy among modules

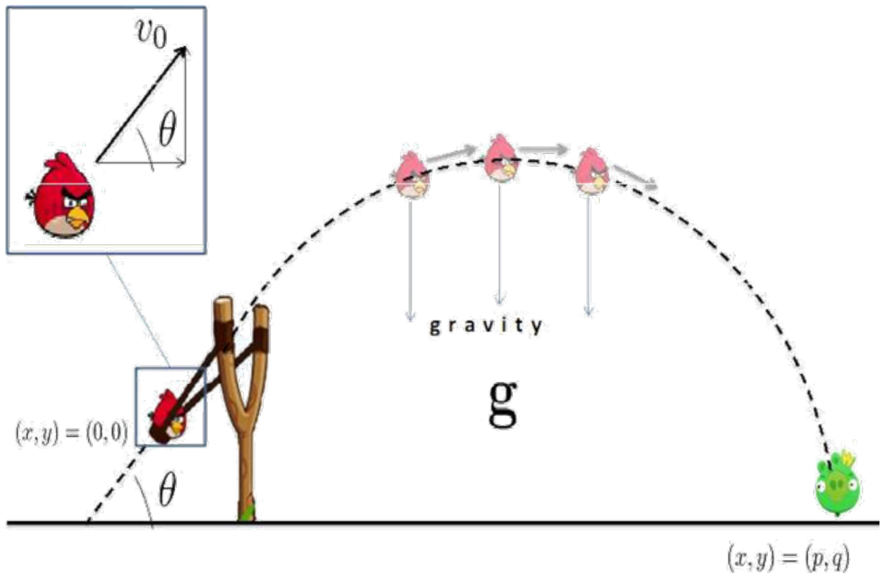

Figure 4: The image demonstrates a physics problem using 'Angry Birds', showing a bird being launched from a slingshot at an angle θ with initial velocity v_o, aiming to hit targets A and B placed at different heights and distances.

This image 5 is a simple graphical representation of projectile motion at different angles. It illustrates three separate trajectories for an object launched from a catapult at angles of 70°, 45°, and 20° above the horizontal. The height and range of each trajectory vary with the angle of launch. The 45° angle is commonly known to give the maximum range under ideal conditions without air resistance. Each trajectory is marked with points that likely represent equal time intervals, demonstrating how the angle affects both the time of flight and the horizontal distance traveled by the projectile. This kind of diagram is typically used in physics education to teach the principles of projectile motion.

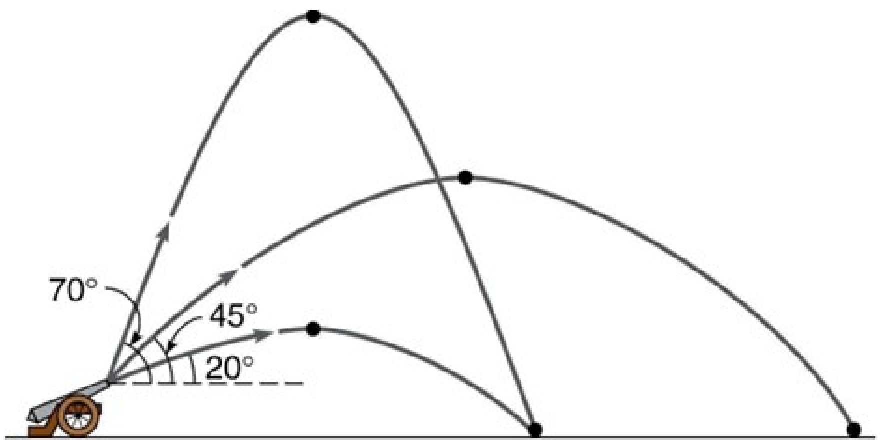

Figure 5: The image shows trajectories of projectiles launched at different angles -20°, 45°, and 70° to illustrate the range of motion in physics, with the 45° angle likely providing the greatest range.