Module Interface Specification for Physics Game Based on Angry Birds

Al Jubair Hossain

March 15, 2024

1 Revision History

Date	Version	Notes
March 5, 2024	1	Changed section (Hyperlink)
March 13, 2024	2	Changed section (5)

Contents

2 Introduction

2.1 Purpose

The purpose of this Module Interface Specification (MIS) document is to define the interfaces for the various modules within the physics-based gaming application. It aims to ensure a clear understanding of how different modules interact and the specifications they must adhere to for successful integration.

2.2 Scope

This document covers the interface specifications for each module involved in the physics-based gaming application, including components for physics simulation, user controls, graphics rendering, sound effects, and other relevant functionalities.

3 Overall System Architecture

3.1 High-Level System Overview

The physics-based gaming application is designed with a modular and scalable architecture to provide an engaging gaming experience. The high-level system overview outlines the primary components and their interactions.

1. **Physics Engine:** Responsible for simulating realistic collision dynamics, gravitational forces, and additional forces. It interfaces with the Projectile and Target/Obstacle modules.

2. **User Interface:** Allows users to interact with the game, providing controls for setting initial conditions, launch parameters, and adjusting the time step size. It interfaces with the User Controls module.

3. **Graphics and Sound Engine:** Visualizes physics-based interactions and implements sound effects. It interfaces with the Physics Engine, User Controls module, and the Projectile and Target/Obstacle modules for rendering scenes and triggering sounds.

4 Flow of Execution:

1. User inputs through the User Interface, setting initial conditions and launch parameters.

2. The User Controls module processes input and communicates with the Physics Engine to simulate the projectile's motion.

3. The Physics Engine updates the positions and velocities of the projectile and targets based on collision dynamics, gravitational forces, and additional forces.

4. The Graphics and Sound Engine renders the updated scene, providing a visual representation of physics interactions and triggering sound effects.

4.1   Key Components

4.1.1   Projectile Module

- **Responsibilities:** Represents the projectile with properties like position, velocity, and mass.

- **Interactions:** Interfaces with the Physics Engine and User Controls module.

4.1.2   Target/Obstacle Module

- **Responsibilities:** Manages targets and obstacles with various shapes and sizes.

- **Interactions:** Interfaces with the Physics Engine for collision checks.

4.1.3   Physics Engine

- **Responsibilities:** Simulates collision dynamics, gravitational forces, and additional forces.

- **Interactions:** Interfaces with the Projectile and Target/Obstacle modules, as well as the User Controls module.

4.1.4   User Controls Module

- **Responsibilities:** Handles user inputs for setting initial conditions, launch parameters, and time step adjustments.

- **Interactions:** Interfaces with the Physics Engine and Graphics and Sound Engine.

4.1.5   Graphics and Sound Engine

- Responsibilities: Visualizes the game scene and implements sound effects.

- Interactions: Interfaces with the Physics Engine, User Controls module, and Projectile and Target/Obstacle modules.

4.2 System Dependencies

The components within the system have dependencies to ensure proper communication and functionality:

1. Physics Engine Dependencies

   - Receives input from the Projectile and Target/Obstacle modules.
   - Provides output to the User Controls module for visualization and sound effects.

2. User Controls Module Dependencies:

   - Sends user input to the Physics Engine to adjust simulation parameters.
   - Interfaces with the Graphics and Sound Engine for rendering and sound effects.

3. Graphics and Sound Engine Dependencies:

   - Receives updates from the Physics Engine for rendering scenes.
   - Interfaces with the User Controls module for user interactions.
   - Triggers sound effects based on events from the Physics Engine.

4. Projectile and Target/Obstacle Module Dependencies

   - Provide input to the Physics Engine for collision checks and trajectory calculations.

5 Module Interface Specifications

5.1 Physics Engine Module

The Physics Engine Module is responsible for simulating realistic collision dynamics, gravitational forces, and additional forces within the physics-based gaming application.

5.1.1 Functions

simulate Physics (projectile, targets, additional Forces)
   Description: Simulates the physics interactions between the projectile and targets, considering gravitational forces and any additional forces.

Parameters:

- **projectile:** Object representing the projectile with properties like position, velocity, and mass.

- **targets:** Array of objects representing targets or obstacles with properties like position, shape, and size.

- **Additional Forces:** Array of additional forces acting on the system.

Output:

- Updates the positions and velocities of the projectile and targets based on collision dynamics, gravitational forces, and additional forces.

Check Collision (projectile, targets)
Description: Checks for collisions between the projectile and targets.
Parameters:

- **projectile:** Object representing the projectile with properties like position, velocity, and mass.

- **targets:** Array of objects representing targets or obstacles with properties like position, shape, and size.

Output:

- Returns information about any collisions detected, including the objects involved and the collision points.

apply Gravity(projectile)

Description:

Applies gravitational forces to the projectile.

Parameters:

- **projectile:** Object representing the projectile with properties like position, velocity, and mass.

Output:

- Updates the projectile's velocity and position based on gravitational forces.

5.1.2 Dependencies

- **Dependent Modules:**

  - Interfaces with the Projectile Module for obtaining information about the projectile.
  - Interfaces with the Target/Obstacle Module for obtaining information about targets or obstacles.
  - Interfaces with the User Controls Module for adjusting simulation parameters.

- **External Dependencies:**

  - May depend on external libraries or frameworks for mathematical calculations related to collision dynamics and gravitational forces.

5.1.3 Exceptions

- **Simulation Exception:**

  - Raised if there are issues during the physics simulation process.
  - May include errors related to invalid input parameters or unexpected behaviors.

- **Collision Exception**

  - Raised if there are issues during collision detection.
  - May include errors related to invalid input parameters or unexpected behaviors.

- **Gravity Exception:**

  - Raised if there are issues during the application of gravitational forces.
  - May include errors related to invalid input parameters or unexpected behaviors.

5.2 User Interface Module

The User Interface Module is responsible for handling user inputs, providing controls for setting initial conditions, launch parameters, and adjusting the time step size in the physics-based gaming application.

5.2.1 Functions

Set Initial Conditions (position, velocity, mass)
  Description: Allows the user to set the initial conditions for the projectile.
  Parameters:

- **position:** The initial position of the projectile.

- **velocity:** The initial velocity of the projectile.

- **mass:** The mass of the projectile.

Output:

- Updates the initial conditions for the projectile.

Set Launch Parameters (angle, force)

Description:

Enables the user to specify the launch angle and force for the projectile.

Parameters:

- a**ngle:** The launch angle of the projectile.

- **force:** The launch force applied to the projectile.

Output:

- Updates the launch parameters for the projectile.

Adjust Time Step Size (time Step)

Description: Allows the user to adjust the time step size for simulation accuracy.

- Updates the time step size for the physics simulation.

5.2.2   Dependencies

- **Dependent Modules:**

  - Interfaces with the Physics Engine Module to transmit user-specified parameters and adjustments.

5.2.3   Exceptions

- **Input Exception:**

  - Raised if there are issues with user input parameters.
  - May include errors related to invalid input values or unexpected behaviors.

- **Adjustment Exception:**

  - Raised if there are issues during the adjustment of simulation parameters.
  - May include errors related to invalid input values or unexpected

  behaviors.

6 Auxiliary Constants

6.1 Physical Constants

Physical constants are essential values that remain constant throughout the physics-based gaming application. They are used for accurate simulations and calculations.

6.1.1 Gravitational Constant

- **Symbol:** G
- **Value:** Approximately 6.674×10^{-11} N (m^2) / $\left(kg^2\right)$
- **Description:** Represents the gravitational force constant.

6.1.2 Mass of the Earth

- **Symbol**: M_E
- **Value**: Approximately 5.972×10^{24} kg
- **Description:** Represents the mass of the Earth.

6.1.3 Acceleration Due to Gravity on Earth

- **Symbol:** g
- **Value:** Approximately 9.81 m/s^2
- **Description:** Represents the acceleration due to gravity on Earth.

6.2 Game Parameters

Game parameters define the characteristics and behaviors of the physics-based gaming application. They influence gameplay and the visual representation of the game.

6.2.1 Maximum Launch Force

- **Symbol:** MAX_LAUNCH_FORCE
- **Value:** 1000 N
- **Description:** Represents the maximum force that can be applied during projectile launch.

6.2.2 Maximum Launch Angle

- **Symbol:** MAX_LAUNCH_ANGLE

- **Value: 90** degrees

- **Description:** Represents the maximum launch angle for the projectile.

6.2.3 Default Time Step Size

- **Symbol:** DEFAULT_TIME_STEP

- **Value:** 0.01 s

- **Description:** Represents the default time step size for the physics simulation.

6.3 Environmental Constants

Environmental constants are factors related to the virtual environment in which the physics-based gaming application operates.

6.3.1 Air Density

- **Symbol:** AIR_DENSITY

- **Value:** 1.225 kg/m^3

- **Description:** Represents the air density in the virtual environment.

6.3.2 Wind Speed

- **Symbol:** WIND_SPEED

- **Value:** 0 m/s (default, no wind)

- **Description:** Represents the speed of the wind in the virtual environment.

6.3.3 Coefficient of Restitution

- **Symbol:** COR

- **Value:** 0.8

- **Description:** Represents the coefficient of restitution, determining the elasticity of collisions.

7 Verification Activities

7.1 Requirements Verification

Verification of requirements ensures that the physics-based gaming application meets the specified criteria outlined in the problem statement. This process involves validating that each requirement is correctly implemented and that the overall system fulfills its intended purpose.

7.1.1 Traceability Matrix

A traceability matrix is a tool used to link each requirement back to its source and forward to the components or modules responsible for its implementation. It aids in ensuring that all requirements are addressed and helps in tracking changes.

Example Traceability Matrix:

Requirement ID	Description	Implemented By
REQ-1	Implement Realistic collision dynamics	Physics Engine Module
REQ-2	Integrate gravitational forces	Physics Engine Module
REQ-3	Allow user specification of launch angle	User Interface Module collision
REQ-4	Implement sound effects for collision	Graphics and Sound Module

7.1.2 Inspection and Review Procedures

Inspection and review procedures involve thorough examinations of the design documents, code, and other artifacts to identify potential issues and ensure compliance with requirements.

Procedure: Design Document Review

- Review the Module Interface Specification (MIS) documents for each module.

- Verify that each module's interface aligns with the specified requirements.

- Check for clarity, completeness, and consistency in the design documentation.

Procedure: Code Review

- Examine the code for each module, ensuring adherence to the specified interfaces.

- Verify that the code implements the desired physics interactions, user controls, and graphics and sound effects.

- Identify and address any deviations from the design specifications.

Procedure: User Interface Testing

- Test the user interface module to ensure proper functionality of user controls.

- Confirm that users can set initial conditions, specify launch parameters, and adjust the time step size successfully.

- Identify and address any issues related to user interaction.

Procedure: Integration Testing

- Conduct integration tests to verify the seamless interaction between modules.

- Ensure that modules exchange data correctly and produce the expected outputs.

- Address any integration issues or inconsistencies.

8 Module Interface Specification (MIS)

8.1 Physics Engine Module

8.1.1 Functions

8.1.1.1 simulate Physics (projectile, targets, additional Forces) Description: Simulates the physics interactions between the projectile and targets, considering gravitational forces and any additional forces.

Parameters:

- **projectile:** Object representing the projectile with properties like position, velocity, and mass.

- **targets:** Array of objects representing targets or obstacles with properties like position, shape, and size.

- **Additional Forces:** Array of additional forces acting on the system.

Output:

- Updates the positions and velocities of the projectile and targets based on collision dynamics, gravitational forces, and additional forces.

8.1.1.2 Check Collision (projectile, targets)

Description:

Checks for collisions between the projectile and targets.

Parameters:

- **projectile:** Object representing the projectile with properties like position, velocity, and mass.

- **targets:** Array of objects representing targets or obstacles with properties like position, shape, and size.

Output:

- Returns information about any collisions detected, including the objects involved and the collision points.

8.1.1.3 Apply Gravity (projectile)

Description:

Applies gravitational forces to the projectile.

Parameters:

- projectile: Object representing the projectile with properties like position, velocity, and mass.

Output:

- Updates the projectile's velocity and position based on gravitational forces.

8.1.2 Dependencies

- **Dependent Modules:**

  - Interfaces with the User Interface Module for adjusting simulation parameters.
  - Interfaces with the Graphics and Sound Module for visualization and sound effects.
  - Interfaces with the Game Logic Module for overall game state management.

- **External Dependencies:**

  - May depend on external libraries or frameworks for mathematical calculations related to collision dynamics and gravitational forces.

8.1.3 Exceptions

- **Simulation Exception:**

  - Raised if there are issues during the physics simulation process.
  - May include errors related to invalid input parameters or unexpected behaviors.

- **Collision Exception:**

  - Raised if there are issues during collision detection.
  - May include errors related to invalid input parameters or unexpected behaviors.

- **Gravity Exception:**

  - Raised if there are issues during the application of gravitational forces.
  - May include errors related to invalid input parameters or unexpected behaviors.

8.2 User Interface Module

8.2.1 Functions

6.2.1.1 set Initial Conditions (position, velocity, mass)

Description: Allows the user to set the initial conditions for the projectile.

Parameters:

- **position:** The initial position of the projectile.

- **velocity:** The initial velocity of the projectile.

- **mass:** The mass of the projectile.

Output:

- Updates the initial conditions for the projectile.

6.2.1.2 set Launch Parameters (angle, force)

Description: Enables the user to specify the launch angle and force for the projectile.

Parameters:

- **angle:** The launch angle of the projectile.

- **force:** The launch force applied to the projectile.

Output:

- Updates the launch parameters for the projectile.

8.2.1.1 Adjust Time Step Size (time Step)

Description:

Allows the user to adjust the time step size for simulation accuracy.

Parameters:

- Time Step: The new time step size for the physics simulation.

Output:

- Updates the time step size for the physics simulation.

8.2.2 Dependencies

- **Dependent Modules:**

  - Interfaces with the Physics Engine Module to transmit user-specified parameters and adjustments.
  - Interfaces with the Graphics and Sound Module for user interactions.

8.2.3 Exceptions

- Input Exception:

  - Raised if there are issues with user input parameters.
  - May include errors related to invalid input values or unexpected behaviors.

- Adjustment Exception:

  - Raised if there are issues during the adjustment of simulation parameters.
  - May include errors related to invalid input values or unexpected behaviors.

8.3 Graphics and Sound Module

8.3.1 Functions

8.3.1.1 Render Scene (projectile, targets) Description: Renders the current scene with the updated positions of the projectile and targets.

Parameters:

- projectile: Object representing the projectile with updated properties.

- targets: Array of objects representing targets or obstacles with updated properties.

Output:

- Visualizes the physics-based interactions in the game.

8.3.1.2 Play Sound(event)

8.3.2 Description:

Triggers sound effects corresponding to key physics events.

Parameters:

- **event:** The physics event (e.g., collision, launch) for which to trigger the sound effect.

Output:

- Plays the appropriate sound effect.

8.3.3 Dependencies

- **Dependent Modules:**

  - Interfaces with the Physics Engine Module for scene updates.
  - Interfaces with the User Interface Module for event triggers.

8.3.4 Exceptions

- **Rendering Exception:**

  - Raised if there are issues during the rendering process.
  - May include errors related to graphical rendering or unexpected behaviors.

- **Sound Exception:**

  - Raised if there are issues during the sound playback process.
  - May include errors related to sound file loading or unexpected behaviors.

8.4 Game Logic Module

8.4.1 Functions

8.4.1.1 manage Game Flow ()

8.4.2 Description:

Manages the overall flow of the game, including starting, pausing, and ending game sessions.

Parameters:

- None

Output:

- Controls the progression of the game based on user interactions and physics events.

8.4.2.1 Update Game State ()

Description:

Updates the internal game state based on user actions and physics events. Parameters:

- None

Output:

- Adjusts the game state, such as scoring, based on user interactions and physics events.

8.4.3 Dependencies

- Dependent Modules:

  - Interfaces with the Physics Engine Module for physics-related events.
  - Interfaces with the User Interface Module for user interactions.

8.4.4 Exceptions

- **Game Flow Exception:**

  - Raised if there are issues during the management of the game flow.
  - May include errors related to unexpected game state transitions.

- **Game State Exception:**

9 Physics motion and use

The Image in Figure 1 illustrates a bird from "Angry Birds" being stretched on a slingshot, with vectors indicating the force of the spring (F_{spring}) pulling upwards and the force of gravity (F_{grav}) pulling down wards, as well as the displacement (s) of the stretch.

9.1 Angle of projection
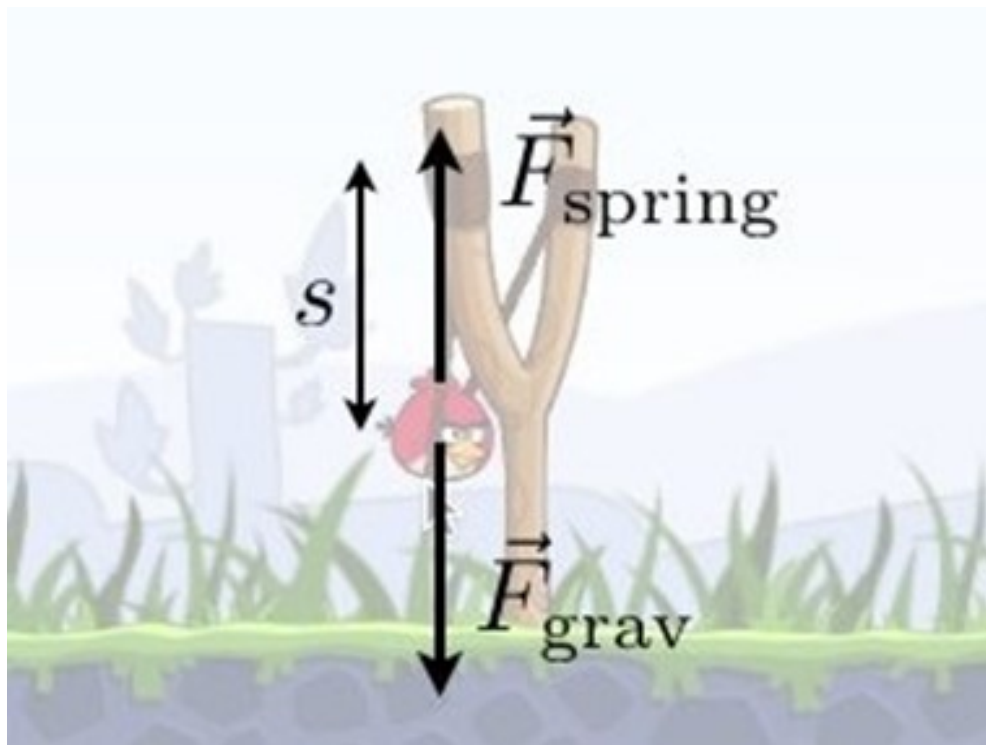
Figure 1: "Angry Bird stretched on a slingshot"

Figure 2 displays a bird from 'Angry Birds' in a slingshot, with notations for the stretch distance (s), the launch angle θ, and the vertical height (h) to conceptualize the physics of projectile motion.

Figure 3 demonstrates a physics problem using 'Angry Birds', showing a bird being

16
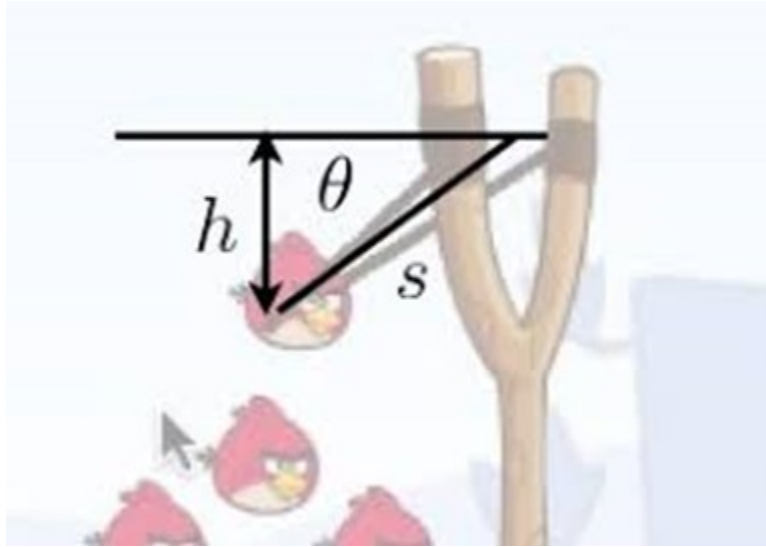
Figure 2: Physics of projectile motion for a slingshot

launched from a slingshot at an angle with initial velocity vo, aiming to hit targets A and B
placed at different heights and distances.
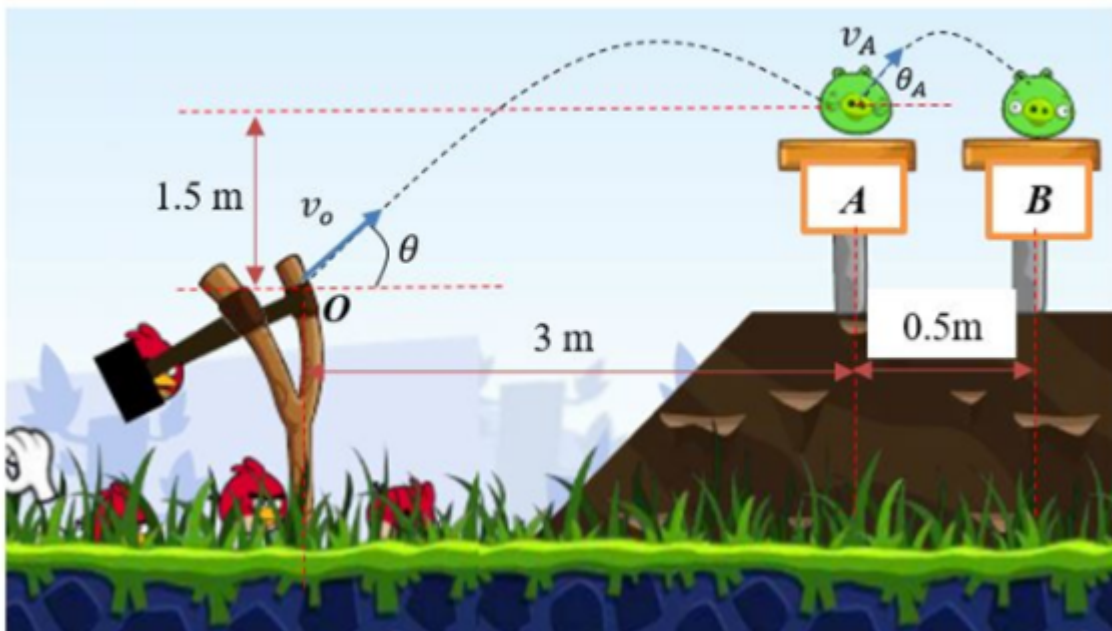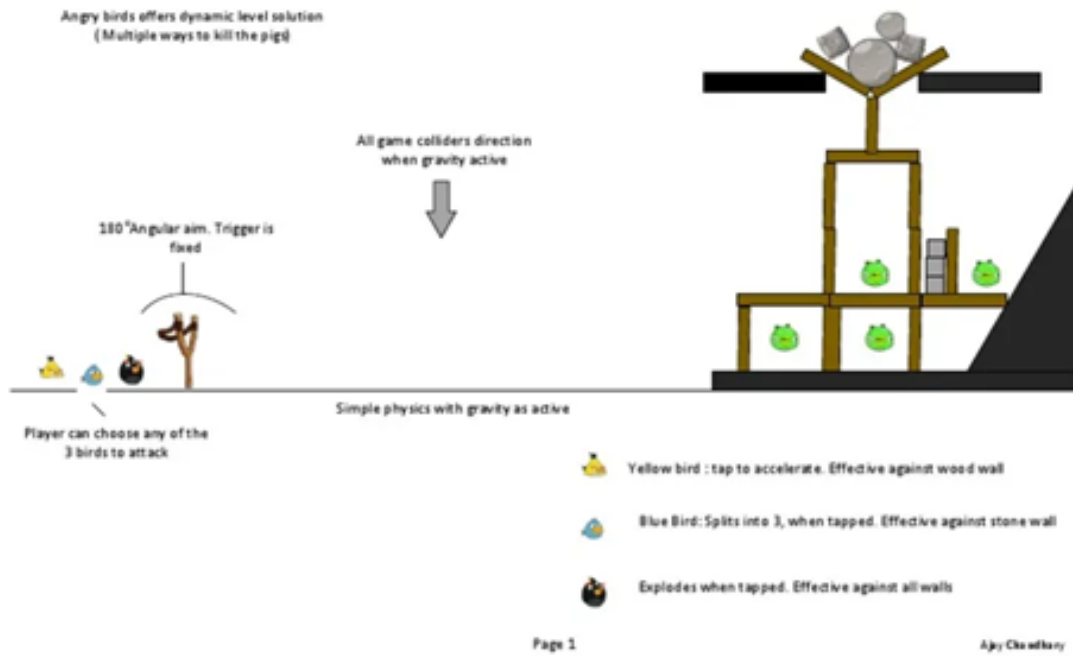

Figure 3: Slingshot Aimed at targets A and B

9.2 Collision

Figure 4: Caption provided in image

Angry Birds Example Level 1
Simple Level

Angry birds offers dynamic level solution
(Multiple ways to kill the pigs)

All game colliders direction
when gravity active

180°Angular aim. Trigger is
fixed

Simple physics with gravity as active

Player can choose any of the
3 birds to attack

Yellow bird : tap to accelerate. Effective against wood wall

Blue Bird: Splits into 3, when tapped. Effective against stone wall

Explodes when tapped. Effective against all walls

Ajay Chaudhary

Solution that offers high
score

BOOM

Fig 1

Fig 2
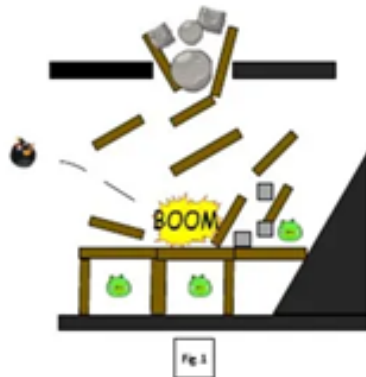
Figure 1
Player can select the default pawn to destroy the
tower. Explosion will remove the support from big
rocks above. They will come crashing down. This bird
gives more chance of destroying all pigs in one go.

Figure 2
Player can select the yellow bird to destroy the wall
supporting the rocks. Once the support is destroyed
rocks will come crashing down. This bird gives more
chance of better score.

Backup if player unable to
destroy objective in 2

Figure 5: Caption provided in image