Coin Flip Simulator - Technical Documentation

Project Overview

This Godot 4.5 project implements a physics-based coin flip simulator that allows users to spawn up to 20 coins, each with randomized positions and rotations. The physics engine determines whether each coin lands on HEADS, TAILS, or EDGE based on its final orientation after coming to rest.

Architecture Overview

Scene Hierarchy

```
Project Root
├── coin_physics.tscn (Main Scene)
│   ├── Main (Node3D) - Root with coin_manager.gd
│   ├── Camera3D
│   ├── DirectionalLight3D
│   ├── Ground (StaticBody3D)
│   │   ├── MeshInstance3D
│   │   └── CollisionShape3D
│   └── UI (CanvasLayer)
│       ├── ThrowButton (Button)
│       └── ResultsDisplay (RichTextLabel)
│
└── coin.tscn (Prefab/Template)
    └── Coin (RigidBody3D) - with coin.gd
        ├── MeshInstance3D (Cylinder)
        └── CollisionShape3D (Cylinder)
```

Script Architecture

1. **coin_manager.gd** - Orchestrates the entire simulation
2. **coin.gd** - Individual coin behavior and physics detection
3. **main.gd** - Legacy/unused script (can be deleted)

Detailed Component Breakdown

1. Main Scene (coin_physics.tscn)

Purpose: The primary game scene that contains the environment, UI, and manager logic.

Key Components:

- **Main Node (Node3D)**
  - Script: `coin_manager.gd`
  - Role: Manages coin spawning, tracks coin count, displays results

- **Camera3D**
  - Position: (0, 3, 5)
  - Rotation: ~-28.6° on X-axis
  - View: Angled downward to see the ground plane

- **DirectionalLight3D**
  - Provides scene lighting

- **Ground (StaticBody3D)**
  - Physics Material:
    - Bounce: 0.3
    - Friction: 0.5
  - Mesh: Plane/Box shaped
  - Purpose: Surface for coins to land on

- **UI (CanvasLayer)**
  - **ThrowButton**
    - Position: (20, 20)
    - Size: (150, 50)
    - Text: "Throw Coins"
  - **ResultsDisplay (RichTextLabel)**
    - Position: (20, 70)
    - Shows coin flip results as they complete

---

2. Coin Scene (coin.tscn)

Purpose: Reusable coin template that gets instantiated for each flip.

RigidBody3D Properties:
- **Mass:** 1.0 kg
- **Gravity Scale:** 1.0 (normal gravity)
- **Physics Material:**
  - Bounce: 0.5 (was 0.7 initially)
  - Friction: 0.5 (was 0.3 initially)
- **Contact Monitor:** Enabled
- **Max Contacts:** 1

Visual Representation:
- **MeshInstance3D:** CylinderMesh
  - Height: 0.1 units (thin like a real coin)
  - Radius: 0.5 units

Physics Collision:
- **CollisionShape3D:** CylinderShape3D
  - Height: 0.1 units
  - Radius: 0.5 units
  - Matches mesh dimensions perfectly

Script Documentation

coin_manager.gd

Purpose: Central controller for the coin flip simulation.

Key Variables:
```gdscript
@onready var throw_button = $UI/ThrowButton
@onready var results_display = $UI/ResultsDisplay
@onready var coin_scene = preload("res://coin.tscn")
```

```
var coin_count = 0
const MAX_COINS = 20
var results_text = ""
```

Flow Diagram:
```
User clicks "Throw Coins" button
    ↓
_on_throw_button_pressed()
    ↓
Check if coin_count < MAX_COINS
    ↓
spawn_coin()
    ↓
1. Increment coin_count
2. Instantiate coin from coin.tscn
3. Add as child to scene
4. Assign coin_id
5. Connect to coin's flip_result signal
6. Call coin.start_flip()
    ↓
Wait for coin to settle...
    ↓
_on_coin_result(coin_id, result)
    ↓
Update ResultsDisplay with result
```

Key Methods:

1. **`_ready()`**
   - Connects button signal
   - Initializes results display with header

2. **`_on_throw_button_pressed()`**
   - Validates coin limit
   - Calls spawn_coin()

3. **`spawn_coin()`**
   - Instantiates new coin from template
   - Assigns unique coin_id
   - Connects to coin's signal
   - Triggers the flip

4. **`_on_coin_result(coin_id, result)`**
   - Receives result via signal
   - Updates UI with formatted result string
   - Prints to console

---

coin.gd

Purpose: Individual coin behavior - handles physics, randomization, and result detection.

Signal:

```gdscript
signal flip_result(coin_id: int, result: String)
```

Key Constants:
```gdscript
const VELOCITY_THRESHOLD = 0.1           # m/s
const ANGULAR_VELOCITY_THRESHOLD = 0.1   # rad/s
const STOPPED_TIME_REQUIRED = 0.3        # seconds
```

State Variables:
```gdscript
var is_flipping = false
var stopped_timer = 0.0
var result_determined = false
var coin_id = 0
@export var heads_face_axis = Vector3.UP  # Which face is "heads"
```

Core Algorithm: Coin Flip Detection

Phase 1: Randomized Spawn
```gdscript
func randomize_spawn():
    # Position randomization (view-centered)
    X: randf_range(-1.5, 1.5)   # Horizontal spread
    Y: randf_range(4.0, 7.0)    # Variable drop height
    Z: randf_range(-2.0, 1.0)   # Depth in camera view

    # Rotation randomization (mostly camera-facing)
    X: randf_range(-45, 45)     # Forward/back tilt
    Y: randf_range(-180, 180)   # Full spin
    Z: randf_range(-30, 30)     # Slight roll
```

Design Rationale:
- X/Z ranges keep coins visible in camera frame
- Y range creates visual variety without excessive fall time
- Rotation ranges ensure coins tumble realistically
- Y rotation (spin) has full 360° freedom

Phase 2: Motion Monitoring
```gdscript
func _physics_process(delta):
    if not is_flipping:
        return

    # Check velocities
    var lin_vel = linear_velocity.length()
    var ang_vel = angular_velocity.length()

    if lin_vel < THRESHOLD and ang_vel < THRESHOLD:
        stopped_timer += delta
```

```
        if stopped_timer >= STOPPED_TIME_REQUIRED:
            determine_flip_result()
            result_determined = true
            is_flipping = false
    else:
        stopped_timer = 0.0  # Reset if still moving
```

Design Rationale:
- Monitors both linear AND angular velocity (coin might spin without moving)
- Requires sustained stillness (0.3s) to avoid false positives
- Resets timer if motion detected (prevents premature determination)

Phase 3: Result Determination
```gdscript
func determine_flip_result():
    # Transform local "heads" axis to world space
    var heads_direction = global_transform.basis * heads_face_axis

    # Compare with world up vector
    var dot_product = heads_direction.dot(Vector3.UP)

    # Classify result
    if dot_product > 0.5:
        result = "HEADS"
    elif dot_product < -0.5:
        result = "TAILS"
    else:
        result = "EDGE"

    flip_result.emit(coin_id, result)
```

Mathematical Explanation:

The **dot product** between two unit vectors returns:
- `+1.0`: Vectors point same direction
- `0.0`: Vectors perpendicular
- `-1.0`: Vectors point opposite directions

Thresholds:
- `> 0.5`: Heads face is >60° toward up (HEADS)
- `< -0.5`: Heads face is >60° toward down (TAILS)
- Between: Coin on edge or angled

Why 0.5 threshold?
- `cos(60°) ≈ 0.5`
- Allows 30° tolerance from perfectly vertical
- Realistic edge case detection

Signal Architecture (Event Flow)

```
User Input
    ↓
```

```
Button.pressed signal
    ↓
coin_manager._on_throw_button_pressed()
    ↓
Spawn coin & call start_flip()
    ↓
coin.randomize_spawn() → Physics begins
    ↓
coin._physics_process() → Monitoring loop
    ↓
Coin stops moving
    ↓
coin.determine_flip_result()
    ↓
coin.flip_result signal emitted
    ↓
coin_manager._on_coin_result()
    ↓
Update UI
```

Signal Connection:
```gdscript
# In coin_manager.spawn_coin():
coin.flip_result.connect(_on_coin_result)
```

This creates a **one-to-many** relationship:
- One manager listens to many coins
- Each coin reports independently when done
- Asynchronous design allows multiple coins to flip simultaneously

---

Physics Configuration

Why These Values?

Property	Value	Reasoning
Coin Bounce	0.5	Realistic metallic bounce without excessive energy
Coin Friction	0.5	Enough slide for natural motion, but settles quickly
Ground Bounce	0.3	Less bouncy than coin creates dampening effect
Ground Friction	0.5	Helps coins slow down after impact
Coin Mass	1.0 kg	Standard unit for easy physics tuning
Gravity Scale	1.0	Normal Earth gravity

Interaction Effects:
- Coin bounces more than ground → realistic rebound
- Similar friction values → predictable sliding
- Low mass → responsive to physics without excessive momentum

---

Design Patterns & Best Practices

1. Prefab/Template Pattern
- `coin.tscn` acts as a template
- `instantiate()` creates independent copies

- Each instance maintains own state

2. Signal-Based Communication
- Decouples coin logic from manager
- Scalable to any number of coins
- Manager doesn't poll - coins push updates

3. State Machine (in coin.gd)
```
States:
- Waiting (is_flipping = false)
- Flipping (is_flipping = true)
- Completed (result_determined = true)
```

4. Threshold-Based Detection
- Velocity thresholds prevent jitter
- Time threshold prevents false positives
- Dot product threshold handles orientation

Possible Enhancements

Short Term:
1. **Visual Polish:**
   - Add heads/tails texture to coin
   - Particle effects on impact
   - Sound effects (clink, bounce)

2. **UX Improvements:**
   - "Clear All" button to reset coins
   - Coin counter display
   - Statistics (total heads vs tails)

3. **Input Options:**
   - Number slider to throw multiple coins at once
   - Keyboard shortcut for throwing

Medium Term:
1. **Advanced Physics:**
   - Air resistance for more realistic arcs
   - Variable gravity modes
   - Different coin materials (gold, silver, copper)

2. **Gameplay Elements:**
   - Best-of-N mode
   - Betting system
   - Multiplayer coin flip battles

3. **Visualization:**
   - Slow-motion replay
   - Camera follow coin option
   - Trail effect during flight

Long Term:
1. **3D Models:**
   - Real coin models (penny, quarter, etc.)
   - Custom coin designer

```
   - Historical coins

2. **Physics Playground:**
   - Adjustable gravity slider
   - Custom bounce/friction controls
   - Wind simulation

---

## **Known Limitations**

1. **Edge Cases:**
   - Coin can theoretically balance on edge indefinitely
   - No timeout mechanism if coin gets stuck

2. **Performance:**
   - 20 coin limit is arbitrary
   - Many simultaneous physics bodies may slow older hardware

3. **UI:**
   - Results display has no scrolling
   - Can overflow with many results

4. **Cleanup:**
   - Coins persist forever once spawned
   - No automatic cleanup or reset

---

## **Testing Checklist**

- [x] Single coin spawn and detection
- [x] Multiple coin spawning
- [x] Result accuracy (heads vs tails)
- [x] Edge case detection
- [x] UI updates correctly
- [x] 20 coin maximum enforced
- [x] Signal connections work
- [x] Random positioning stays visible
- [ ] Performance with 20 coins
- [ ] Long-term stability test

---

## **File Structure Summary**

```
CoinToss/
├── coin.gd              # Individual coin logic
├── coin.tscn            # Coin prefab
├── coin_manager.gd      # Main controller
├── coin_physics.tscn    # Main scene
├── main.gd              # (Unused - can delete)
├── icon.svg             # Godot project icon
└── project.godot        # Project settings
```

```

Quick Start Guide

1. Open `coin_physics.tscn`
2. Press F5 to run
3. Click "Throw Coins" button
4. Watch physics simulation
5. Results appear in UI as coins settle

Document Version: 1.0
Godot Version: 4.5 stable
Last Updated: October 11, 2025