**Double Pendulum Simulation Documentation**

**Introduction**

This project implements a 3D double pendulum simulation using the Godot engine. The simulation demonstrates the programmatic movement of a physical system in 3D space, featuring two pendulum arms in a hierarchical relationship. Running at 60 frames per second through Godot's "\_physics\_process" method, the simulation provides smooth, real-time movement while approximating realistic pendulum physics.

**Git Repository Link:** <https://github.com/BIRKARAN-SINGH/Double-Pendulum>

**Video Demonstration:** <https://drive.google.com/file/d/1C93f2MUIuu2sUmEy_9dHbOY1USNV7vUf/view?usp=sharing>

**Objectives**

* Create a realistic 3D double pendulum simulation using hierarchical object modeling
* Implement smooth 60 FPS updates using "\_physics\_process"
* Apply basic physics principles for approximate pendulum motion
* Develop a user-interactive system with pause and speed control features
* Ensure proper camera tracking and visualization of the pendulum movement

**Technical Implementation**

**Core Components**

**1. Main Scene (main.tscn)**

The main scene establishes the foundational structure with these key elements:

* RootNode (Node3D) - Controls overall simulation logic
* Camera3D - Provides user viewpoint with slight offset for better observation
* WorldEnvironment - Manages lighting and visual environment
* Pivot1 (SpringArm3D) - First pendulum arm anchor point
* Hierarchical pendulum structure (Arm1 -> Bob1 -> Pivot2 -> Arm2 -> Bob2)

**2. Scripts and Logic**

**Main Script (Main.gd)**

@export var angle1: float = 45.0 *# Initial angle for first arm*

@export var angle2: float = -30.0 *# Initial angle for second arm*

@export var speed1: float = 3.5 *# Angular velocity for first arm*

@export var speed2: float = 4.2 *# Angular velocity for second arm*

@export var amplitude1: float = 50.0 *# Swing amplitude for first arm*

@export var amplitude2: float = -40.0 *# Swing amplitude for second arm*

A screen shot of a computer

Description automatically generated

Key features:

* Configurable initial angles and speeds
* Amplitude control for swing range
* Pause functionality
* Speed adjustment system

**Pivot Control (pivot\_1.gd)**

Manages the fixed position of the pendulum's anchor point and ensures proper arm following behavior.

**User Controls and Input Events**

**Key Bindings**

* **Pause Control**:
  + Key: K
  + Action: Toggles the simulation pause state
  + Implementation: Configured in project.godot input map as "pause" event
* **Speed Control**:
  + Keys: Up/Down Arrow keys
  + Actions:
    - Up Arrow: Increases pendulum speed
    - Down Arrow: Decreases pendulum speed
  + Speed Increment: 0.5 units per key press
  + Implementation: Configured in project.godot input map as "up" and "down" events

The input system is implemented through Godot's built-in Input Map system, providing reliable key event handling and easy configuration. Speed adjustments are constrained to prevent negative values, ensuring stable simulation behavior.

A screen shot of a computer program

Description automatically generated

**Testing Results**

**1. Functionality Testing**

**Motion Consistency Test**

**Test Case**: Verify smooth pendulum movement at different speeds

* **Initial Speed**: Speed1 = 3.5, Speed2 = 4.2
* **Results**:
  + Smooth sinusoidal motion observed
  + No visible stuttering at default speeds
  + Consistent amplitude maintenance

**Pause Functionality Test**

**Test Case**: Verify pause system behavior

* **Method**: Toggle pause multiple times during operation
* **Results**:
  + Instant response to pause command
  + Maintains exact position during pause
  + Smooth continuation after unpause

**Speed Control Test**

**Test Case**: Test speed adjustment system

* **Method**: Increment and decrement speed using arrow keys
* **Results**:
  + Proper speed adjustment with 0.5 unit steps
  + No negative speed values possible
  + Smooth transition between speeds

**2. Physics Simulation Testing**

**Pendulum Synchronization Test**

**Test Case**: Verify proper interaction between pendulum arms

* **Method**: Observe relative motion of both arms
* **Results**:
  + Second arm properly inherits first arm's motion
  + Maintains correct pivot point relationships
  + Amplitude values properly affect swing range

**Position Tracking Test**

**Test Case**: Verify accurate position updates for both pendulum arms

* **Method**: Monitor debug output of pivot rotations and bob positions
* **Results**:
  + Accurate position tracking through debug logs
  + Proper maintenance of arm lengths
  + Correct hierarchical transformations

**3. Performance Testing**

**Frame Rate Stability Test**

**Test Case**: Verify consistent 60 FPS operation

* **Method**: Run simulation with debug monitoring
* **Results**:
  + Maintains target 60 FPS
  + No significant frame drops during normal operation
  + Smooth performance across speed ranges

**Memory Usage Test**

**Test Case**: Monitor resource utilization during extended operation

* **Method**: Run simulation for 30+ minutes
* **Results**:
  + Stable memory usage
  + No memory leaks detected
  + Consistent performance over time

**Known Limitations**

1. Physics simplification may not perfectly match real pendulum behavior
2. Fixed anchor point cannot be moved during simulation
3. Speed adjustments are linear and may not maintain perfect physical accuracy

**Future Improvements**

1. Add realistic physics calculations based on actual pendulum equations
2. Implement dynamic anchor point positioning
3. Add trajectory tracking visualization
4. Include energy conservation monitoring
5. Add additional visual effects for enhanced user experience

**Conclusion**

The double pendulum simulation successfully implements a visually engaging and functionally accurate representation of pendulum motion in 3D space. The hierarchical structure ensures proper motion inheritance, while the user controls provide interactive engagement with the simulation. Testing results demonstrate stable performance and reliable functionality across all key features.

**Project Development Process**

**1. Development Meetings and Milestones**

| **Date** | **Attendees** | **Topics Discussed** | **Outcomes** |
| --- | --- | --- | --- |
| Dec 1, 2024 | Birkaran, Fuzail | Initial project planning, task distribution | Defined project structure, assigned primary responsibilities |
| Dec 3, 2024 | Birkaran, Fuzail | Physics implementation review | Reviewed pendulum equations, adjusted amplitude parameters |
| Dec 5, 2024 | Birkaran, Fuzail | Implementation of user controls | Completed pause and speed control features |
| Dec 8, 2024 | Birkaran, Fuzail | Visual enhancement and testing | Refined camera positioning, added lighting effects |
| Dec 9, 2024 | Birkaran, Fuzail | Documentation and final testing | Completed documentation, performed final system tests |
| Dec 10, 2024 | Birkaran, Fuzail | Documentation review and submission preparation | Finalized all components, prepared submission package |

**2. Device Specifications**

| **Specification** | **Birkaran's Device** | **Fuzail's Device** |
| --- | --- | --- |
| Operating System | Windows 11 Home | Windows 11 Home |
| CPU Processor | 11th Gen Intel i5-11300H | Intel Core i5-1035G1 |
| RAM | 16 GB | 24.0 GB (23.7 GB usable) |

**2. Contributors and Responsibilities**

**Birkaran Singh**

* Core development and physics implementation
* Pendulum motion calculations
* Input system implementation
* Performance optimization

**Fuzail Chaugle**

* Documentation and visual enhancement
* Camera system implementation
* Testing coordination
* Developed environment for double pendulum

**Version Information**

* Godot Engine Version: 4.3
* Project Name: COMP360\_Assignment#3