**Doble Pendulum**

**Introduction**

In this assignment, the primary goal is to create a 3D simulation of a double pendulum, demonstrating the programmatic movement of a physical system in a 3D space. This system consists of at least two objects structured in a hierarchical relationship to represent the pendulum arms, allowing for a realistic swinging motion. The simulation will run within the Godot game engine, utilizing the “\_physics\_proces” method to update object positions at a default rate of 60 frames per second, ensuring smooth, real-time movement. Although perfect physics accuracy is not required, the project encourages the use of basic physics principles to approximate realistic motion. This assignment also emphasizes collaborative testing and refinement, offering hands-on experience with 3D animation techniques, hierarchical object control, and real-time physics simulation in game development.

**Link for the Video:** [**https://drive.google.com/file/d/1pUSSBSiDNIOKm8eNce4lcPGjL\_GXeJSn/view?usp=drive\_link**](https://drive.google.com/file/d/1pUSSBSiDNIOKm8eNce4lcPGjL_GXeJSn/view?usp=drive_link)

**Objective**

* Develop a 3D simulation of a double pendulum using hierarchical objects to mimic realistic swinging motion.
* Implement the “\_physics\_process” method to update the pendulum's position at 60 frames per second for smooth animation.
* Utilize basic physics principles to model approximate motion of a double pendulum in a 3D space.
* Ensure the system operates within Godot’s constraints, using educationally licensed objects if necessary, up to 200 MB.
* Conduct collaborative testing to refine movement, ensuring stable and visually coherent performance.

**Development Process**

1. Double Pendulum Hierarchical Structure:

* Object Hierarchy Creation: Design a two-object hierarchy to represent the double pendulum arms, ensuring they are parented correctly for accurate motion inheritance between them.
* 3D Model Integration: Import 3D models for the pendulum arms, staying within the 200 MB limit and utilizing educationally licensed models as required.

1. Pendulum Motion Simulation:

* Physics-based Motion Calculation: Approximate double pendulum motion using simplified physics equations to simulate the swinging effect.
* Real-time Updates: Use the “\_physics\_process” method to update object positions at 60 frames per second, ensuring smooth movement in the 3D space.
* Angle and Rotation Adjustments: Apply rotation calculations to each pendulum arm based on hierarchical motion, mimicking realistic joint dynamics.

1. Testing and Calibration:

* Simulation Testing: Conduct iterative testing to adjust parameters, refine motion smoothness, and ensure stability within the Godot engine.
* Group Collaboration: Collaborate with team members to evaluate the system’s performance, focusing on achieving consistent movement without excessive oscillations or glitches.

1. Optional Visual Effects:

* Basic Visual Enhancements: Add optional visual effects like shadows or simple particle trails to emphasize pendulum movement.
* Environment Setup: Create a basic scene for context, ensuring the pendulum system remains the visual focus of the simulation.

**Project structure**

1. Scene Configuration:

* Main Scene (double\_pendulum\_root.tscn): Initializes the primary scene for the double pendulum simulation, setting up the hierarchical structure of pendulum arms, camera, and environment. It links necessary scripts for updating pendulum motion and handling visual effects.

1. Scripting:

* PendulumController Script (pendulum\_controller.gd):
* Manages the hierarchical motion of the double pendulum by calculating rotational angles and updating each arm's position in real-time.
* Uses physics-based calculations to approximate pendulum swing and update positions within the “\_physics\_process” method.
* PhysicsSimulation Script (physics\_simulation.gd):
* Controls the physical parameters for the pendulum, including gravity, damping, and swing velocity.
* Configures simplified physics equations to produce a dynamic, natural swinging effect in 3D space.
* CameraController Script (camera\_controller.gd):
* Follows the pendulum’s motion, providing a fixed or orbiting view based on user input.
* Adjusts view angles and distance to keep the pendulum as the focal point.

**Implementation Details:**

1. Pendulum Hierarchical Structure and Motion:

* Hierarchy Setup: Establish a two-level hierarchy by parenting the second pendulum arm to the first, enabling the inherited motion of the second arm based on the first’s movement.
* Physics-based Motion: Calculate approximate angular velocity and acceleration for each pendulum arm, using simplified equations to simulate gravitational and inertial effects on the swinging motion.
* Real-time Position Updates: Implement the “\_physics\_process” function to update pendulum positions 60 times per second, allowing real-time simulation of the double pendulum dynamics.\

1. Pendulum Swing Animation:

* Position Interpolation: Use interpolation between calculated angles at each frame to ensure smooth transitions for both pendulum arms.
* Rotation Calculation: For each arm, calculate its rotation based on its angle and position within the hierarchy. This maintains a natural swinging motion, emphasizing the chaotic and dynamic behavior characteristic of a double pendulum.

1. Camera Tracking and Dynamic View:

* Smooth Camera Tracking: Implement a function that follows the double pendulum’s center of mass smoothly, creating a stable view that dynamically adjusts to the pendulum’s motion.
* Orbit Controls: Add user input options to orbit the camera around the pendulum, offering various angles to observe the swinging and enhancing the immersive experience.

**Testing Approach:**

1. Motion Testing: Verify the pendulum’s swinging motion to ensure smooth transitions and realistic behavior of both arms in the hierarchy. Observe if each arm exhibits expected dynamics based on the double pendulum model.
2. Hierarchy Adjustment: Test the relationship between the two pendulum arms to confirm correct inheritance of motion, ensuring that the second arm’s motion is influenced by the first arm’s position and swing.
3. Physics Parameter Calibration: Experiment with various gravity, damping, and angular velocity settings to identify the optimal balance between realistic motion and system stability.
4. Camera Tracking and Controls: Check that the camera follows the pendulum system smoothly, maintaining a steady view of the motion. Test the orbit controls to confirm that users can adjust the camera angle dynamically.

**Conclusion:**

This assignment provided a hands-on experience with 3D simulation techniques by developing a double pendulum system in the Godot engine. By simulating hierarchical motion and approximate physics-based dynamics, the project fostered a deeper understanding of animation control, real-time updates, and smooth object movement in a 3D space. The addition of camera tracking and optional visual effects enhanced the simulation, demonstrating the importance of both technical precision and visual clarity in creating an engaging experience.

The project successfully achieved a balance between simplified physics and real-time motion, highlighting the double pendulum’s complex, chaotic behavior in a manageable form. Potential future improvements might include incorporating more detailed physics, additional pendulum arms, or user interaction to expand the educational and visual impact. Overall, this assignment built a foundational understanding of 3D motion simulation and prepared us for more advanced projects in game and simulation development.

**Contribution:**

Birkaran Singh:

Fuzail Chaugle: