**2D Physics Library for Games Programming**

**Thirulogeswaren V1, Mrs. Umadevi G2**

1Student, Computer Science and Engineering, Agni College of Technology, Chennai, Tamil Nadu, India

2Assistant Professor, Computer Science and Engineering, Agni College of Technology, Chennai, Tamil Nadu, India

**ABSTRACT**

This paper is about designing and implementing a simple 2D physics library that lets video game programmers create physics bodies that are influenced by forces and can collide with other physics bodies in an environment in a physically plausible manner. The library is written using C++17 and STL with no external dependencies, just using the core features that come with the language. The library aims to be simple, lightweight, and flexible, which can be used alongside other development libraries or rendering engines for creating video games.

**KEYWORDS:** physics library, video game programming, physics bodies, C++17, STL, 2D, development libraries, rendering engines.

**1. INTRODUCTION**

Physics Engines are complex software that is responsible for simulating physical systems and interacting with other systems in a virtual environment, underneath the system they use extensive mathematical algorithms for computing the motion of physical object which is based on Newton’s Laws of Motion, to detect if two or more objects are intersecting or touching and to resolve them such that they stop intersecting or penetrating into each other, at the end of the day it’s just a big calculator that does the mathematics in order to simulate physics.

Physics Engines are prominent games system in the Video Games Industry, as they allow us to experience real-world simulations and physics that are used in some of the modern Game Titles such as Half-Life: Alyx, GTA V, Red Dead Redemption 2, Crysis Remastered, Doom Eternal, as they use their own in-house proprietary physics system and some games use middleware physics library such as NVIDIA’s PhysX, Havok and so on.

**2. LITERATURE SURVEY**

In this Survey, we will look at an overview of current state of research and development in the field of Physics Engines, including their algorithms, simulation methods, and performance optimizations:

[1] Erin Catto - “Box2D: a 2D Physics Engine for Games”, Game Developers Conference (GDC), 2006.

[2] Christopher Hecker - “Physics Engine Performance: A Comprehensive Study”, Game Developers Conference (GDC), 2008.

[3] Erin Catto - “Iterative Dynamics with Temporal Coherence”, Game Developers Conference (GDC), 2005.

[4] Erin Catto - "The Future of Constraint Solvers", Game Developers Conference (GDC), 2011.

[5] Richard Feynman - "Simulating Physics with Computers", International Journal, 1982.

[6] Gino van den Bergen - "Real-Time Collisions and Physics Simulation in Video Games" IEEE Computer Graphics and Applications, 2014.

**3. PROPOSED SYSTEM**

The physics engine discussed here is code named “Debry”, a simple 2D physics library which can be used alongside development libraries such as SDL2, SFML, and Raylib that are used to create 2D video games, applications and much more. The library provides a blueprint to create and manipulate mathematical vectors and common mathematical utility functions that are used in two-dimensional games. Next it comes with a point-mass system that can be influenced by forces and given a geometric shape to check if an object is in collision with other objects in a virtual environment and is reported back to the resolver. This system is implemented as a Particle class and the intersection can be resolved by Particle Resolver method internally. Finally, Debry provides an interface to create custom forces such as gravity, blast force, buoyancy force and more which can be applied to or removed from a particle which updates them internally. Common Forces such as gravity are provided by the library and cannot be extended, the base class interface is known to be as “Force Generator”.

**4. SYSTEM ARCHITECTURE**

**4.1 CORE**

**4.2 POINT MASS SYSTEM**

**4.3 FORCE GENERATORS**

**4.4 DEBRY ARCHITECTURE**

**5. CONCLUSION**

The system discussed in this paper is experimental and not used for commercial purposes. However, there will be a stable release. More features and the current ones will be tested and added to the library as it continues to evolve, and the documentation is still under development.

**6. REFERENCES**

[1] Game Physics Engine Development by Ian Millington.

[2] Real-Time Collision Detection by Christer Ericson.

[3] Game Physics by David H. Eberly.

[4] Custom Math Libraries by Sean Middleditch.