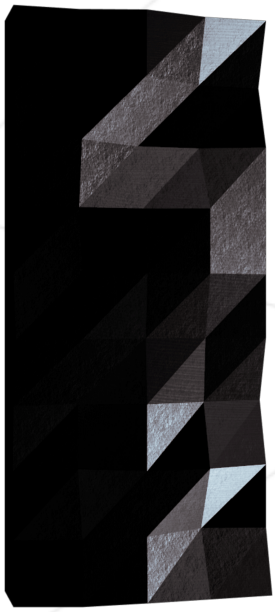

ft_newton

Some Models to describe the world

Summary: In this project, you will code a basic physics engine (like PhysX, Havok, or Bullet Physics Engine)

Contents

Chapter I

Preamble

Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things.
— Sir Isaac Newton

Chapter II

Introduction

This project introduces the physic engine, a part of a classical game engine.
You will learn about collision detection, rigid body, elastic collisions, Torque, inertia tensor, ...
Go as far as you want regarding math and physics but don't take too much time.
Sometimes the best is the enemy of the good.
You'll be able to make a better version of your code in your next project.

Chapter III

Instructions

General instructions :

- You must use OpenGL (v3+) or Vulkan for 3D rendering (your scop/HumanGl should be enough !).
- You can use any 3D Asset, any audio file, free of copyright. The artistic quality is not part of the correction scale...
- You must code the object's interaction, rotation, movement, and collision yourself. No external library for calculation.
- Code in C, C++, or rust.
- Nothing should do physics or collision-related stuff for you, be smart!
- Do you need exception handling, RTTI and such?
- A simple 'make' command should be enough to compile the full project.

Chapter IV

General instructions

You will code small angry birds inspired game :

- A scene where you use either a Catapult or a Trebuchet (this choice is a game-changer) throwing some apples (projectiles) on some unstable composed structures.
- Impact must produce a series of reaction that is physically correct and the system must go back to a stable state. (no infinite bounce)
- You will code at least 3 primitive colliders base on different geometries: a box, a sphere, and a plane.
- The gameplay can be 2D only (like the real angry bird), but your physic and rendering engine must be in 3D.
- It must be easy to spawn a LOT of physical objects on demand.
- An FPS counter and an object counter must be present.
- The scene should be playable with a normal user display without any debugging information.
- A full and direct control of at least: time (reversal is not mandatory), gravity, projectile speed, projectile direction, projectile mass.
- A debug display, that prints the colliders as wireframes, is mandatory and can be turned on/off at any time.
- Everything must be considered as a rigid body (no deformation).
- You will implement the translational motion.
- You will implement the rotational motion (harder you don't need a perfect version here).
- You will implement elastic collisions.
- You will implement gravity.
- You must be able to show every feature in the main game easily (no recompile or 5min setup).

You are free to do anything compatible with those rules.

You need a working simulation that can handle a lot of objects (no brute force!) with basic behavior. Advance physics and simulation are welcome but not mandatory.

Chapter V

Bonus Part

- Big: good ragdoll.
- Medium: A mesh collider that can create an accurate collider based on 'any' 3D models. You need to prepare at least 3 model including 1 humanoid and be prepared to show the precision of the collider with specific testing.
- Small: anything, a nice GUI, better primitive, advanced physics.

Here again, keep big ideas for Very Real Engine.

Chapter VI

Turn-in and peer-evaluation

As usual, turn in your work on your `GiT` repo. Only the work included on the last commit of the master branch will be used during the evaluation.
If your assets are over 1Go replace your asset folder with a clean md5 recursive hash of the folder and bring this folder to your correction.