**Overview:**

**The purpose of this project is to be a robust real-time simulation of 3D rigid bodies with support for concave objects.**

**Libraries:**

**The glm math library is used for the basic vector, matrix and quaternion arithmetic. Various glm objects such as glm::mat3, glm::quat, and glm::vec3 are used quite extensively.**

**The imgui GUI library is being used for all GUIs in engine.**

**The graphics engine was initially written by Nicholas Ammann and is being used with modifications with his permission. The graphics engine is written using OpenGL 3.3. Additionally, it is dependent on stb for image loading, assimp for model loading, glfw 3.2.1 for window management, and glad for OpenGL function bindings.**

**Object Classes:**

**The class representing a 3D rigid body contains the following state information:**

glm::vec3 x; // position

glm::quat q; // orientation

glm::vec3 P; // Linear Momentum

glm::vec3 L; // Angular Momentum

float mass; // Mass

glm::mat3 Ibody; // Inertia tensor of the body

glm::vec3 force; // Net force acting on the body

glm::vec3 torque; // Net torque acting on the body

**This information is sufficient for applying the effects of the forces and torques on the body and integrating the position and orientation of the object. The linear and angular momentum state values can be used to derive our linear and angular velocities. Similarly, the force and torque can be used to derive our linear and angular accelerations. Additionally, the base object contains the following information for debugging and validating state:**

glm::vec3 v; // linear velocity

glm::vec3 w; // angular velocity

glm::vec3 a; // linear acceleration

glm::vec3 wp; // angular acceleration

glm::vec3 cm; // center of mass

**The Rigid Body also keeps a pointer to the model of the object as this is used for the initial inertia calculation. In the future, I plan on using the model to create a collision object. This collision object will contain a convex decomposition of the model (current plan is to calculate using HACD – Hierarchal Approximate Convex Decomposition) along with a bounding box.**

**This collision object will be inserted into a Spatial Partition data structure. Ideally, this Spatial Partition data structure will be either a BSP tree or a K-D tree. The plan is to attempt both and go with the one that is, on average, more performant.**

**Collision Detection:**

**The imgui GUI library is being used for all GUIs in engine.**

**Collision Response:**

**The imgui GUI library is being used for all GUIs in engine.**

**Contact Response:**

**The imgui GUI library is being used for all GUIs in engine.**

**Numerical Integration:**

**The imgui GUI library is being used for all GUIs in engine.**

**References:**

**The imgui GUI library is being used for all GUIs in engine.**