Introduction

To maintain high evaluation standards, you are expected to:

- Remain polite, courteous, respectful, and constructive at every moment of the discussion. Trust between you and our community depends on your behavior.

- Highlight the flaws and issues you uncover in the turned-in work to the evaluated student or team, and take the time to discuss every aspect extensively.

- Please take into account that discrepancies regarding the expected work or functionalities definitions might occur. Keep an open
mind towards the opposite party (is he or she right or wrong?), and grade as honestly as possible. 42's pedagogy only makes sense if peer evaluations are carried out seriously.

Guidelines

- You must grade only what exists in the GiT repository of the student or team.

- Remember to check the GiT repository's ownership: is it the student's or team's repository, and for the right project?

- Check thoroughly that no wicked aliases have been used to trick you into grading something other than the genuine repository.

- Any script supposed to ease the evaluation provided by one party
must be thoroughly checked by the other party to avoid unpleasant situations.

- If the evaluating student hasn't done the project yet, it is mandatory that he or she reads it before starting the evaluation.

- Use the available flags on this scale to tag an empty work, a non-functional work, a coding style ("norm") error if applicable,

cheating, and so on. If a flag is set, the grade is 0 (or -42 in case of cheating).
However, except for the cheating case, you are encouraged to carry on discussing what went wrong, why, and how to address it, even if the grading itself is over.

Attachments

subject.pdf

b266617-f6de-42f7-85

Preliminaries **Basic requirements**

Verify the following points :
- The Makefile must be present and do not relink.
- The program is in C, C++, or Rust.
- The rendering is done using Vulkan, or OpenGL3+ version.

Tests **Basic Tests**

- Outside of big assets it should compile out the box.
- The scene is an "angry birds"-like game.
- The student used a model of Trebuchet and not a weak catapult...Ok, both are fine. - Ask the student how did he implement gravity.
- test with a constant gravity of 0 or other values.
- Play with gravity on and off.
- Play with time per frame.
- during the eval toggle colliders wireframe on and off if you have any doubt.

Basic Tests, Motion, stability

All those features are present and working if not choose FALSE :
-there are (at least) three types of colliders (box, sphere, and plane).
-when you add a lot of objects the simulation continues to run smoothly. -There's a spatial partitioning structure/algorithm to avoid useless collision tests. -After some time, the simulation smoothly goes back to a stable state.
-use slow time to be sure there's no huge glitch on how it's implemented. -overall physic is realistic.

Ask code explanations, how it's done, what data structures are used why etc... You should be able to understand how it's done even if you never did it!

Elastic collisions

-If you collide with an object with a very different mass the collision seems to act properly.
-when two things collide they should bounce, like a sphere falling on the ground.
-Try a very fast projectile or play with the time speed to make a projectile pierce an obstacle without a collision if it's too easy to make it bug choose FALSE.
-The student must explain the basics of collision detection and common bugs associated with it.

Rotation, Inertia tensor, and Torques

Collision creates natural rotations.
from 0 -> things don't rotate.
2 -> spheres rotate realistically. (Not boxes)
3 -> boxes rotate realistically. (Not spheres)
4 -> Both are fine. (but an impact on edges do not trigger bigger rotation forces)
to 5 -> very realistic rotation multiple small impacts create bigger accurate movement and rotation.

Here a quick analogy between translational and rotational motion can help explain what works and how it works. classical Forces make you move Torques make you rotate... Be fair.

Rate it from 0 (failed) through 5 (excellent)

Bonus **Inertia tensor**

Big Bonus a working ragdoll implementation (search on youtube if you don't know how it should look like)

Mesh collider

The mesh collider is implemented, and functional.
Test it with at least three very different models.
This time it's not a box or sphere approximation but a real geometry accurate collision on the 3d model.

Free

Anything that is not mandatory, GUI, multi-thread, SIMD, very clean code... anything it's SHOWTIME

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don’t forget to check the flag corresponding to the defense