Concordia University – Fall 2015
COMP 477 – Animation for Computer Games
Team Snowglobe

Anthony Moniz 26558259
Sebouh Bardakjian 29640185
Francis Hamelin-Owens 27067275

COMP 477 Report

(Objectives)

1. Background reading on the Bullet physics library and set up
2. Set up models, textures of objects (globe and objects within it) for rendering
3. Interactive interface for manipulating the snowglobe
4. Set up rigidbodies and collision boxes for objects with different properties
5. Handle collisions within the snow globe (ball) to contain objects in the ball
6. Generating 'forces' corresponding to user manipulation
7. Create snow particles as sphere colliders rendered as billboards
8. Create wind effects for snow that simulates a snowstorm in a fun way
9. Testing and bug fixing
10. Write documentation and create demo

Introduction

The goal of this project was to create an application with the Bullet Physics library that allows the user to interact with a snow-globe object in order to produce interesting physical responses. The snow globe houses multiple objects and the user has the ability to rotate and shake the snow globe causing objects within it to fall and get knocked around. Many snow particles are simulated and rendered within the snowglobe along with a wind effect to simulate a snowstorm. The project is developed using the Bullet Physics library, C++ and OpenGL library, with support from the glew and glm libraries.

How we handle movement

There are two main objects that the user can interact with, which are the globe and the camera. (*Check Keybinds for all possible actions)* For the camera, we can zoom in and out by translating it in space on the Z axis, we can rotate it around the globe specifically on the X and Y axis, and we can also move the camera on the X and Y axis allowing it to go anywhere. Note that manipulating the camera does not have any effect on the physics, therefore an inverted camera could make it seem like things aren't falling down. This bring us to our next object, the globe, which can also be fully manipulated. The way we handle the globe's movement and rotation is by respectively adding linear velocity, and angular velocity. There are two main ways to move and rotate the globe, which are by mouse or by keyboard keys. We set up a direction vector based on a mouse drag, or choose a constant velocity to apply based on keyboard input. All movement and rotations applied to the globe have an impact on the objects inside the globe, since they may collide, which causes them to move around.

How we simulate snow

Each snowflake is a rigid body with a small spherical collision box. The rigid body is set up so that it has strong dampening forces and a very small angular factor. This minimizes the amount of 'rolling' or sliding around the snowflake does. It also has a lower than average sleeping threshold to minimize unnecessary physics calculation, by taking it out of the equation quickly. Although this does not accurately simulate snow in a real-world sense, it does a fine job of emulating fake snow you would want in a snow globe. The interactions are fun and allow the snow to clump in on the ground or fly around its container.

When ‘wind’ is enabled, each frame a force is applied to each snowflake to push it around. The force a simple sinusoidal wave in all directions relative to each snowflake’s position. Because the force is relative to its position, each snowflake is pushed in its own direction, rather than all snowflakes being blown against one wall of the globe. Although this force function is deterministic, the collisions with the globe and other objects within the globe cause the snow patterns to be unpredictable which leads to a very busy, seemingly-random snowstorm when lots of snowflakes are rendered.

For the rendering, each snowflake is using the same texture and shaders. The vertex shader positions the vertices so that the texture will always face the camera. The fragment shader then applies this information to the texture and also discards if a fragment would be invisible (transparent).

How we created the globe

The snow globe was a fairly complex object to properly set up. Since objects reside inside its sphere, a different approach to simply setting primitive collision shapes was required. One shape which fit our needs was the GImpactMeshShape, which uses the mesh’s triangles to create its collision shape. Since the globe itself was made up of tens of thousands of vertices, the shape was split up into smaller parts. The first part of this collision shape was the sphere in which everything resides. Since the easiest and best way to add collisions to the sphere was with the GImpactMeshShape, another sphere, which approximates the globe’s glass ball, was used and ended up much smaller in terms of triangles. With the sphere completed, the next part was the base where everything would initial rest upon. The base is a simple box shape which extends further outwards than the globe. Since everything resides in the sphere, this could be done without problem, giving us a nice base. The next part was the tree which resides inside. A simple cone shape for the top and cylinder for the base was used to approximate the tree. Finally, all shapes were then combined into a compound shape, thus allowing the globe to be moved as a single object. A linear factor of 0 was also set to remove every velocity applied to the globe which wasn't generated by the user.

Inside the globe, three small presents of different colors are scattered around. These presents use simple box shapes which interact with the globe and snow.

For the rendering, the snow globe and tree are drawn using the information obtained from the collision shapes. The snow globe's glass sphere uses a material with a low alpha value and blueish tint to give a colder vibe to the scene.

End

We were initially interested in doing a project that would allow us to have hands on experience with physics based animation since we are all in the computer games option and found it to be an interesting topic. Working with real-time physics in this project has allowed us to deepen our knowledge and understanding on how to use physics in a small application. Working out how to create forces and how those forces should be applied gave us a better understanding of how forces (actually impulses that are only applied during the physics step) interact with rigid bodies. Simulating hundreds of balls (snowflakes) forced us to see limits of simple collision detecting and handling, and how we can constrain rigidbodies to limit rolling or sliding around. Keeping objects within another object made us try many different approaches to create complex collision shapes, and collision handling techniques that allow for more accuracy.

KEYBINDS

S	Spawn 10 snowflakes in the center of the globe
W	Toggle wind on and off.
H	Toggle collision sphere
G	Toggle globe
F	Toggle collision plane
Q	Quit

Globe	
CTRL + Left mouse button	Rotate the globe
Left mouse button	Move the globe in space
Keyboard Uparrow /Downarrow	Rotate globe around X-axis
Keyboard Leftarrow /Rightarrow	Rotate globe around Y-axis
+	Move the globe up in space
-	Move the globe down in space

Camera	
ALT + Left mouse button	Rotate the camera around the globe (y-axis)
ALT + Right mouse button	Rotate the camera around the globe (x-axis)
ALT + Middle mouse button	Move the camera in space
Scroll Wheel	Zoom in and out

Debugging Toggle (Pick one option)	
V	Remove debugging
B	Draw aabb boxes
N	Draw wireframe
M	Draw aabb & wireframe