Physics-Based Chipmunk2D Game

Detailed Design

Steven Palmer
⟨palmes4⟩
Emaad Fazal
⟨fazale⟩
Chao Ye
⟨yec6⟩

April 26, 2016

Contents

List of Tables

List of Figures

Revision History

Date	Version	Notes
January 10, 2015	1.0	Created document

1 Introduction

This document makes up the second part of the design document. The first part is found in the 'System Architecture' document, which contains a proper introduction for the document as a whole.

2 Module Interface Specification

A module interface specification was prepared using Doxygen. This is included as a separated document entitled 'Detailed Design - Module Interface Specification'.

3 Error Handling

3.1 Missing Files

Several files will be included as part of the game, and will be required for the game to function properly. Required file types include sound files, object mesh/texture files, and shader files. Any missing file will produce a message indicating that the file was not found. The game will continue to run if sound files are not found, but will not play any missing sounds. Missing object or shader files, however, is a more serious error and will result in termination of the application.

3.2 Library Initialization Failures

Library initialization failures, especially GLEW initialization errors, may occur when the game is attempted to be run on a system with outdated hardware. These types of errors will produce an error message followed by termination of the game application. All errors of this type will be considered unrecoverable.

3.3 Unexpected Errors

No other errors are expected to occur given that the game code *should* restrict bad states from occurring. [Get rid of the "should". —DS]

User inputs will be well defined and tested for errors. Other user inputs (i.e. keys that are not used in controlling the game) have no programmed function and as a result should have no effect when pressed. It is still likely, however, that unexpected errors will occur from time to time due to uncaught bugs that rarely execute or external factors (driver error, memory error, etc.). These unexpected errors are not handled and will likely cause the game to crash.

4 User Interface Elements

4.1 User Interface

The game will consist of two general types of user interfaces: menus and game stages/levels. Examples of what these interfaces will look like along with some of their key features are given in the following subsections.

4.1.1 Menu

The main menu of the game will be the level select screen. An example of what this screen will look like is given in Table 1. [Should say "Figure" —DS]

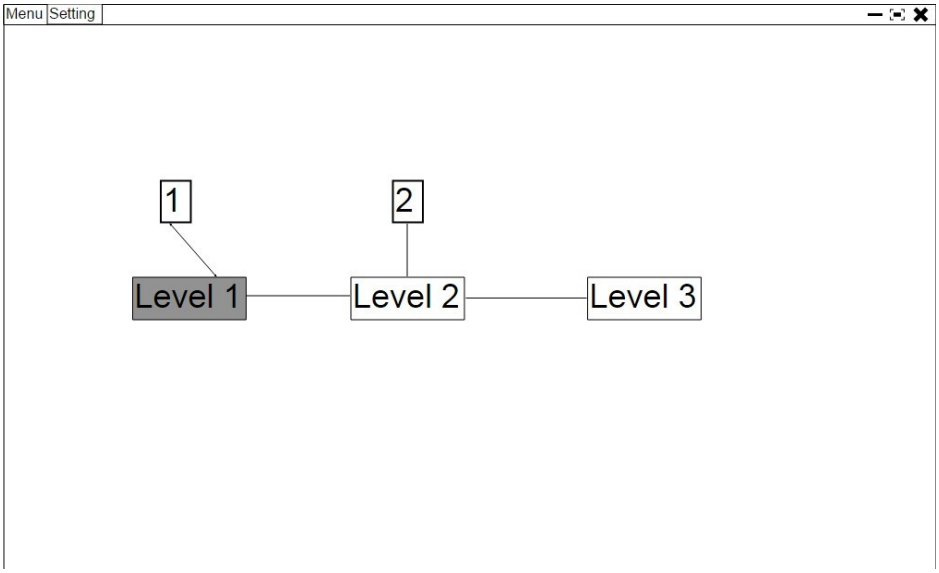

Figure 1: Level select screen.

1. The grey color indicates that this level has already been unlocked. The player can start from this level directly.

2. The white color indicates that this level has not been unlocked yet. The player must start complete the level that precedes it in order to unlock it.

4.1.2 Game Stages

Samples of possible stages are given in Figures 2 - 4. Each figure is used to highlight some of the intended features that will appear in the game.

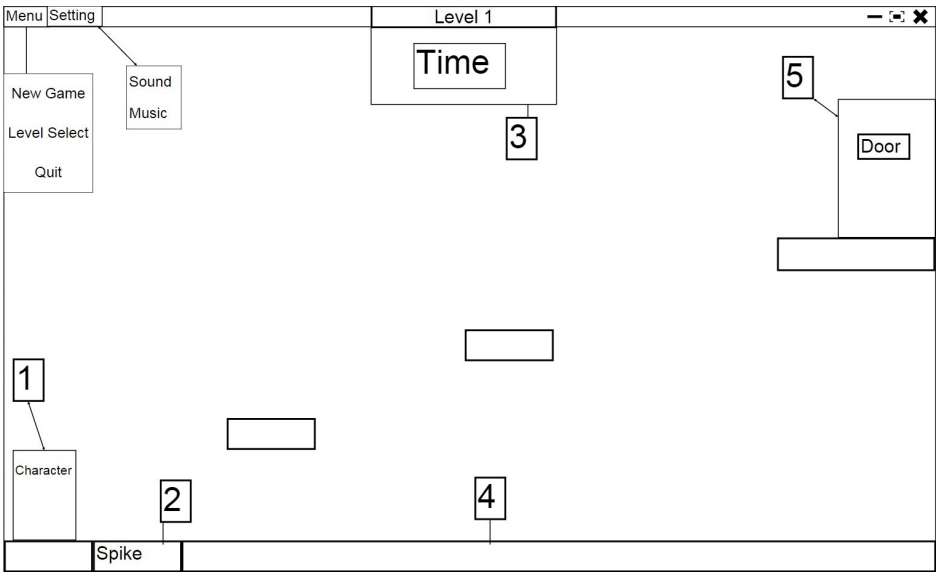

Figure 2: Stage 1 example.

1. The initial position of the hero when the stage starts. When the hero is killed, play will resume at this position.

2. An spike hazard: the hero will be killed if contact is made.

3. The timer increases from zero when the stage begins. The rating achieved when the level is completed is based on the elapsed time.

4. Boundary/platform: the hero can stand/move safely on boundaries/platforms.

5. The goal of the stage is to reach the door safely.

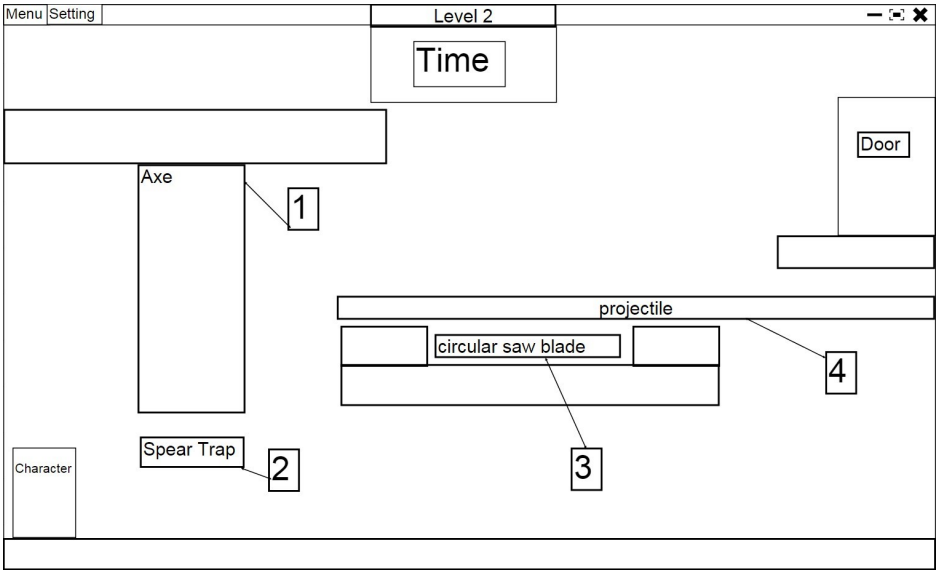

Figure 3: Stage 2 example.

1. A swinging axe hazard: the hero will be killed if contact is made.
2. A spear trap: spears will randomly pop out of the spear trap killing the hero on contact. When the spears are retracted it acts as a platform.
3. A circular saw blade hazard: the saw blade will move back and forth on a linear path continuously, killing the hero on contact.
4. A projectile trap: shoots single spears which kill the hero on contact.

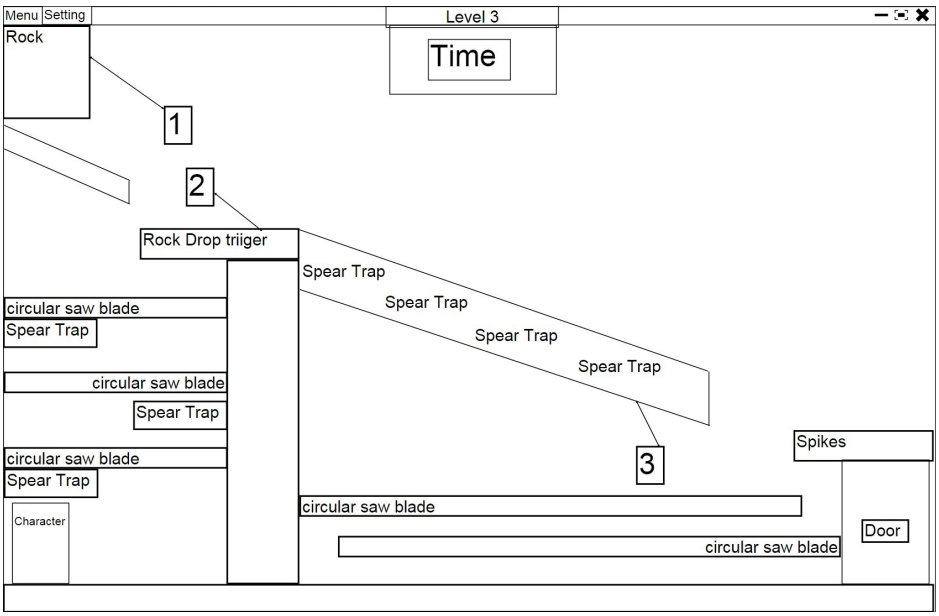

Figure 4: Stage 3 example.

1. Rock trap: the rock drops and falls/rolls freely, crushing the hero if contact is made.
2. Trigger: activates the rock trap (or other types of traps) if touched by the hero.
3. Ice platform: changes friction factors for more difficult control when the hero is in contact.

4.2 Object Meshes

Creating meshes that accurately portray the objects they represent is an important consideration related to the user interface. Samples of object meshes that have been created for use in the game are given in Figures 5 - 12 in Appendix A.

5 Key Algorithms

This section is not applicable to this project.

6 Relational Database Structure

This section is not applicable to this project.

7 Communication Protocols

This section is not applicable to this project.

8 Implementation Specifics

8.1 Language

The project will be implemented using C++. The use of C++ combined with OpenGL will allow for simple and efficient cross platform implementation.

8.2 Supporting Libraries

Libraries that will be used in the implementation of the game are listed in Table 1.

Table 1: Supporting Libraries

Library	Function
Chipmunk 2D	Physics engine
OpenGL	Graphics functionality
OpenAL	Audio functionality
GLFW	Window and IO management
GLEW	Advanced graphics capabilities

9 Appendix A: Mesh Samples

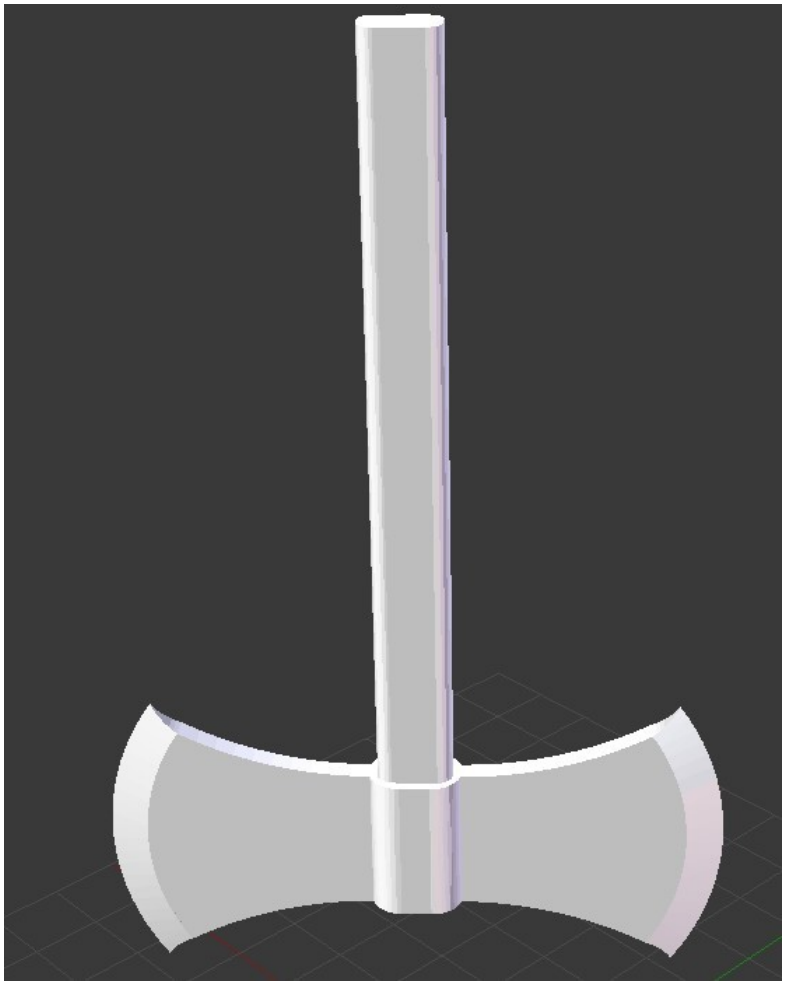

Figure 5: Sample mesh: axe.

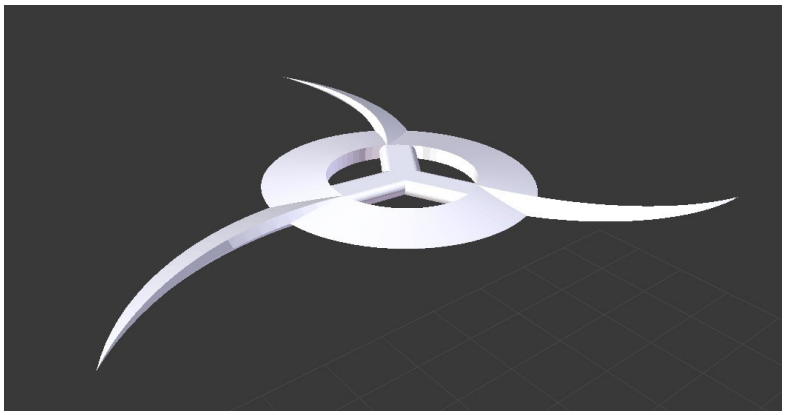

Figure 6: Sample mesh: circular saw blade.

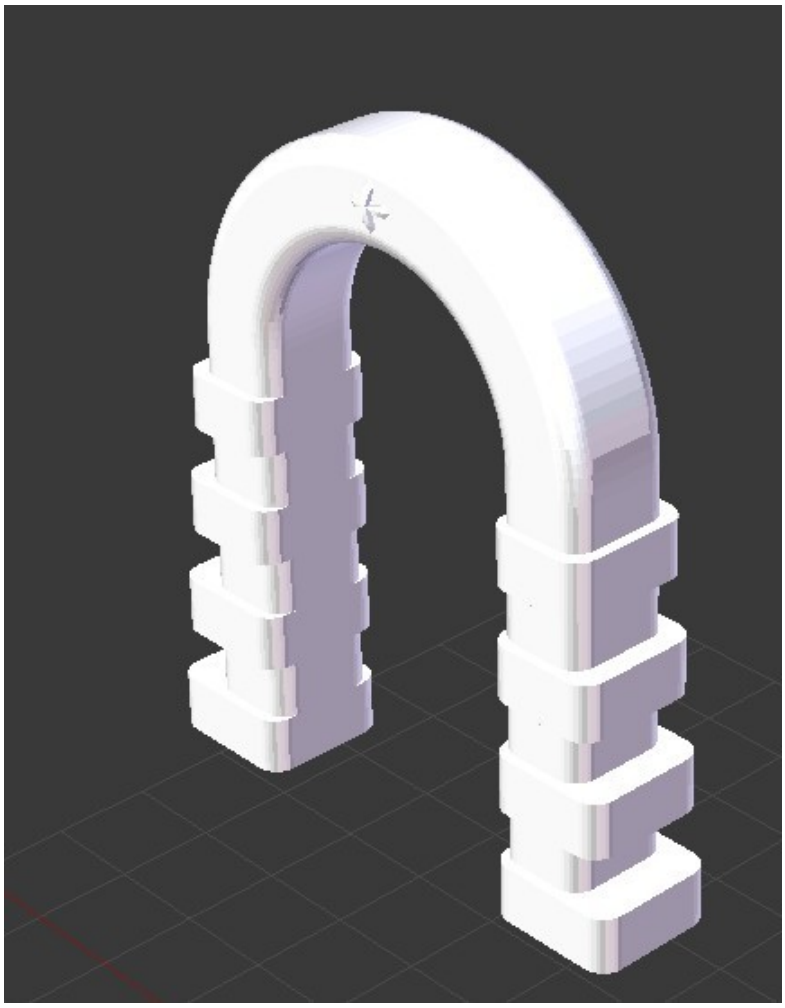

Figure 7: Sample mesh: gate.

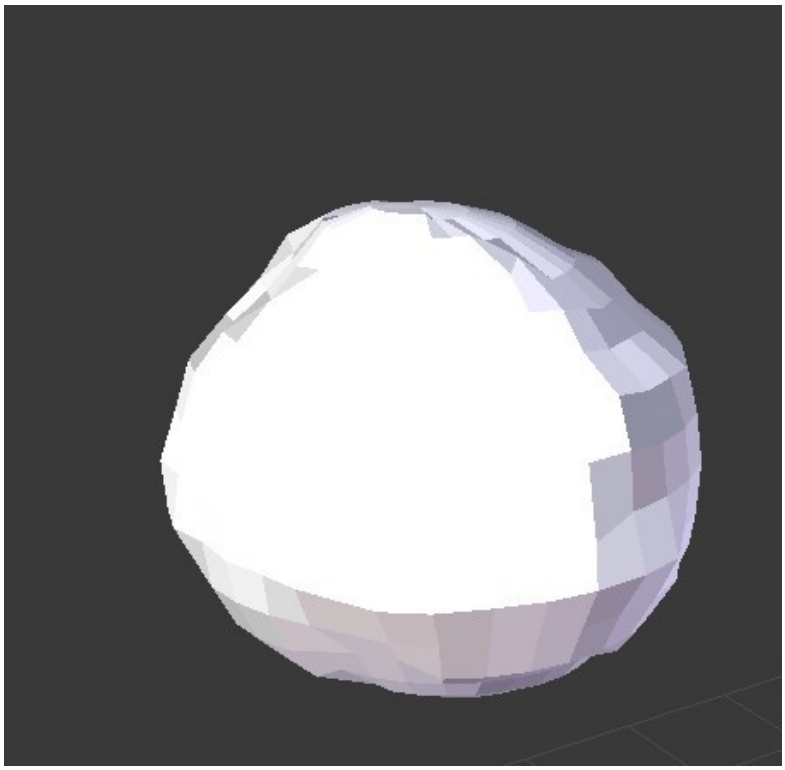

Figure 8: Sample mesh: rock.

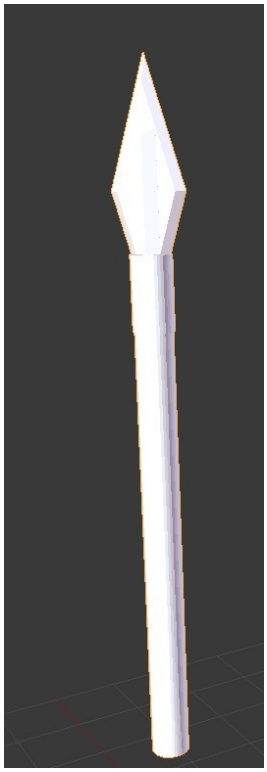

Figure 9: Sample mesh: spear.

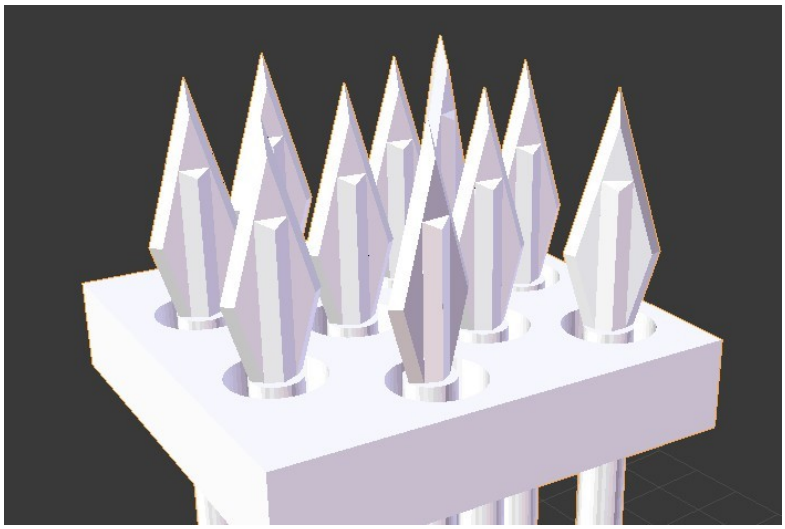

Figure 10: Sample mesh: spear trap.

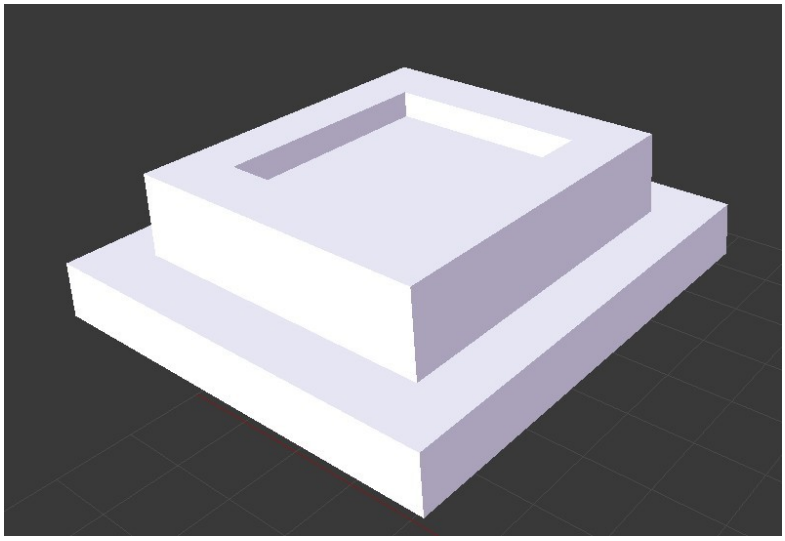

Figure 11: Sample mesh: trigger.

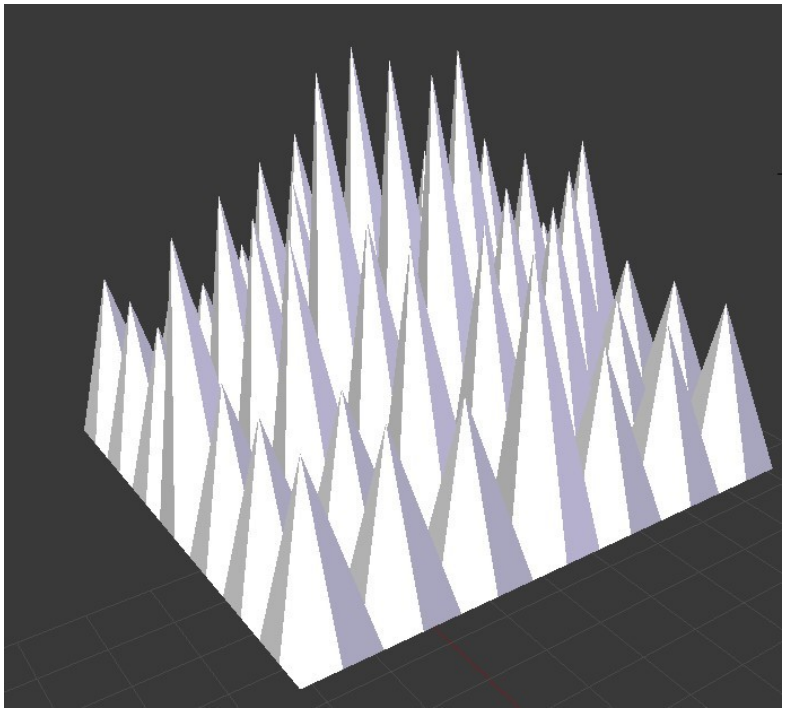

Figure 12: Sample mesh: spikes.