CS201 Data Structures II
Assignment 2

CS Program
Habib University

Spring 2024
Due Date: 10 March 2024 @ 11:59PM

1 Introduction

There are two questions in this assignment. Each question carries 50 marks. Details are as under.

2 Question 1 - (50 marks)

You will apply the knowledge of hashing to implement spatial hashing to resolve particle-particle collisions [1] as shown in Figure 1. Template code has been shared with you which implements particle particle collisions through brute force method. However, the spatial hashing code does not implement spatial hashing for collision detection between particles. You can press 'h' key to toggle from using spatial hashing (default) to brute force method. As you would expect, the frame rate grinds down to a halt when brute force method is used.

For this assignment, you need to add collision detection in the shared template code using spatial hashing. You can get the technical overview of the method using reference [1] and a more implementation friendly description is shared in reference [2]. Relevant sections of the code are marked with TODOs so you can know where you need to add additional code.

If you have implemented spatial hashing based collision correctly, you should see collision detection and response happening at interactive framerates unlike the brute force method which makes the application unusable. Moreover, the UI should display the total collisions detected on screen which in the default template should be 0.

2.1 Setting up raylib

The basic skeleton code is built on top of raylib [3]. You should be able to extract the given zip file and open the folder in VSCode and it should compile out of the box. If you want to get some details on the installation of raylib, the steps to install raylib are given for all platforms separately. For Windows OS, they are given here: `https://github.com/raysan5/raylib/wiki/Working-on-Windows`. For other OS, refer to `https://github.com/raysan5/raylib`. Note for the given code, there is no additional step required. All setup is done already for you. For those of you working on non-Windows OSes, you may refer to the raylib installation steps and get connected with one of the course TAs.

2.2 Running the template code

Unzip the shared template code at your preferred location on your hard drive. Next, go to VSCode and open the template code folder. Press F5 or go to **Run menu** and then select **Start Debugging**. If all goes well, your should see the application run as shown in Figure 2. By default, the code uses spatial hashing with boundary collisions but it does not implement particle-particle collision. The brute force method does the collision both with the boundaries

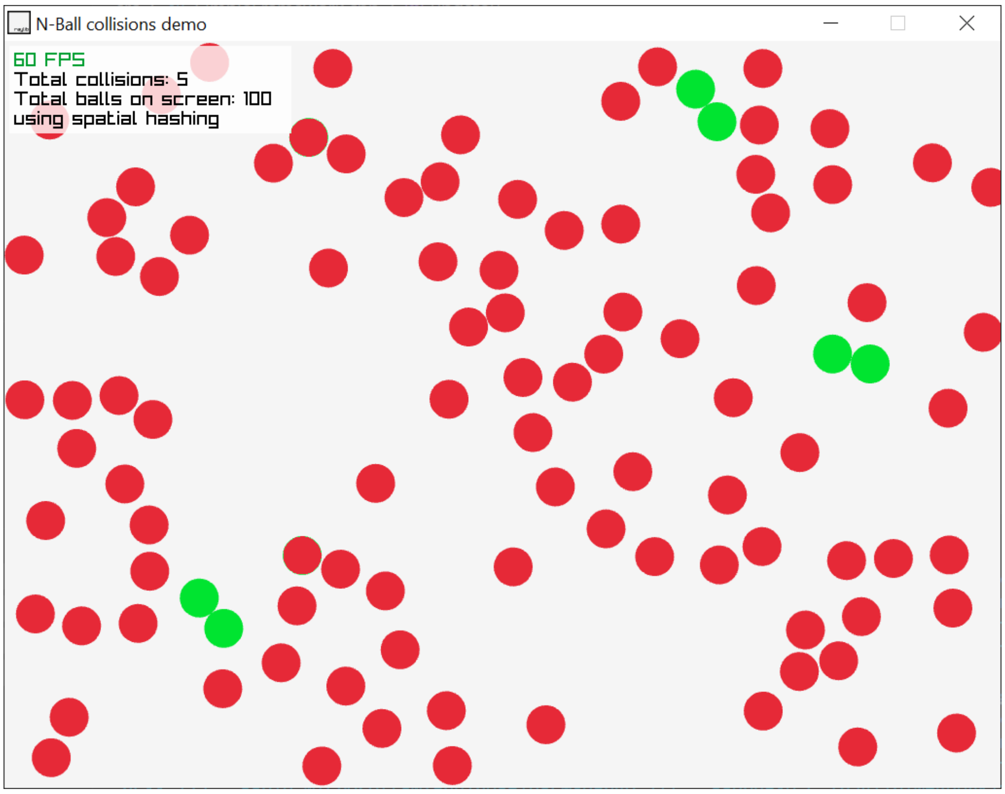


Figure 1: N-particles collision demo.

as well as with all other particles. You can press 'h' key to toggle using spatial hashing to use brute force method. Note that the performance of the application will slow down considerably if you use the brute force method for collision detection.

2.3 Template code framework details

The C++ template code uses a fairly simple C++ framework. There is a base class called **Demo**. This class sets up raylib and the event loop so that you do not have to repeat the steps in your own application. There are four pure virtual functions in the Demo class which are detailed as in Listing 1.

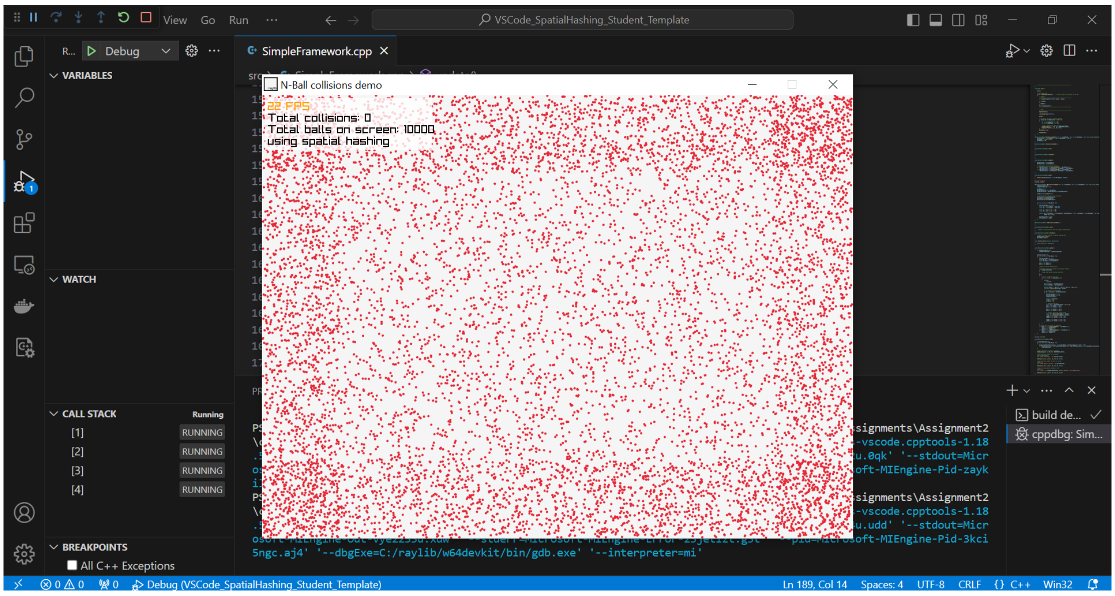
```
virtual void init() = 0;
virtual void shutdown() = 0;
virtual void draw() = 0;
virtual void update() = 0;
```

Listing 1: Demo class pure virtual functions.

The init function is where you would do initialization of your application data structures and other variables. The shutdown method will do the de-initialization that is removing every data structure and data that you will use in our application. The draw function will be where the rendering of the objects will take place and finally, the update function is where the physics simulation will be updated.

2.4 Simple particle collision demo

Within the template framework code, we have given a simple example class inheriting from the Demo class called BallCollisionDemo. This class implements a single particle moving in the window and colliding with the screen bounds. You can see how we override the virtual methods of the Demo class. This is given as a simple example for you to help understand the given Demo



```
22 FPS
Total collisions: 0
Total balls on screen: 10000
using spatial hashing
```

```
5ngc.aj4' '--dbgExe=C:/raylib/w64devkit/bin/gdb.exe' '--interpreter=mi'
```

Figure 2: Running the template code.

framework structure. In order to run this demo, please uncomment line 6 and comment line 7 of main.cpp and then press 'F5' or go to **Run menu** and then **Start Debugging**. If all goes well, you should see a single particle moving in the screen and colliding with the screen bounds as shown in Figure 3.

2.5 N particles collision demo

Within the template framework code, we provide another example class inheriting fom the Demo class called NBallsCollisionDemo. This class implements 10000 particles moving in the window and colliding with the screen bounds. All of the code for initialization and handling of rendering of the 10000 particles is already taken care of for you. In addition, the basic screen collision is also implemented. We provide the brute force inter particle collision code, see lines 187-228 SimpleFramework.cpp but the spatial hashing based optimized inter particle collision detection is missing and this is what you are required to implement in the given framework.

For your assistance, the relevant sections of the code are marked with TODOs comments so you can add relevant code there. Please note that you are free to add code any where you deem fit but please do not remove the code that is already provided. For testing, you can change the last parameter of the NBallsCollisionDemo constructor call in main.cpp to 100 to reduce the total number of particles on screen. This will help you debug the code faster. Once you are fine with the code, dont forget to change the last parameter back to 10000.

2.6 Required Tasks

You should implement a C++ class Hash that should contain all necessary data structures and functions. You should provide at least two functions: a function to repopulate the hash grid given all particle positions and another function to return the list of particles in the hash grid where the given particle position hashes to.

2.7 Rubric

You submission will be evaluated on the following rubric:

1. Code gives the correct output. (+10)
2. Code is commented well using meaningful variable names. (+10)

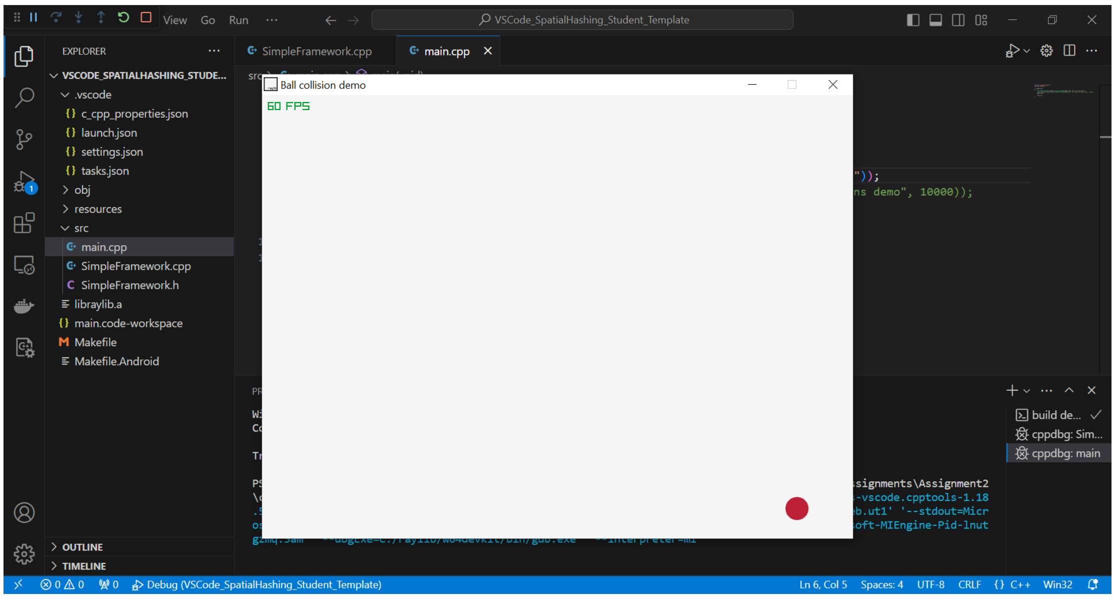


Figure 3: Simple particle demo example.

3. Spatial hashing is correctly implemented. (+30)

2.8 Deliverables

Please test your code to ensure that it performs as expected. Once you are satisfied with it, you can share the updated SimpleFramework.cpp file on the LMS submission module. If there are more than 1 files, please compress all your files into a zip file bearing your team name for e.g. Team1.zip. After submission, please try downloading and extracting the zip file to ensure it works as often times, the shared zip files do not work as expected. Please do not share the template folder with all your files. The source cpp files or any other required .h/.cpp files should be sufficient.

3 Question 2 - (50 marks)

3.1 Problem A - (25 marks)

Let $p = 2^w - 1$ for some positive integer w. Explain why, for a positive integer x,

$$(x\%2^w) + (x/2^w) = x\%(2^w - 1). \tag{1}$$

This gives an algorithm for computing $x\%(2^w - 1)$ by repeatedly setting

$$x = x\&((1 << w) - 1) + x >> w \tag{2}$$

until

$$x <= 2^w - 1. \tag{3}$$

3.2 Problem B - (25 marks)

Consider the simplified version of the code given in Listing 2 for adding an element x to a LinearHashTable, which simply stores x in the first *null* array entry it finds. Explain why this could be very slow by giving an example of a sequence of $O(n)$ add(x), remove(x), and find(x) operations that would take on the order of n^2 time to execute.

```
bool addSlow(T x)
{
   if (2*(q+1) > t.length)
      resize(); // max 50% occupancy

   int i = hash(x);
   while (t[i] != null)
   {
      if (t[i] != del && x.equals(t[i]))
         return false;
      i = (i == t.length-1) ? 0 : i + 1; // increment i
   }
   t[i] = x;
   n++; q++;

   return true;
}
```

Listing 2: LinearHashTable addSlow function definition.

4 Deductions

The following will result in deductions.

1. Code does not compile. (-50)
2. Failure to comply to given instructions. (-20)
3. Late submission less than 24 hrs post deadline (-20)
4. Late submission n days post deadline (-20 - n*10)

Total score = max(0, (total rubric score - total deductions))

5 Using chatGPT or other AI software

You are not allowed to use any AI software to obtain the code for this assignment. If you do use it then you should share the entire chat history with us. If you dont share the chat history and we find that the code has been taken from chatgpt or similar AI software, the case will be reported to the Academic Conduct committee and all members of the group will get a 0 in the assignment.

6 Plagiarism Policy

We have zero tolerance for plagiarism. The assignment submission should be your own genuine work without copying content from anyone else in the class or from the internet. If there is any evidence of plagiarsim, the case will be reported to the Academic Conduct committee and all members of the group will get a 0 in the assignment.

References

[1] Matthias Teschner, Bruno Heidelberger, Matthias Muller, Danat Pomeranets, and Markus Gross, *"Optimized Spatial Hashing for Collision Detection of Deformable Objects"*, in Proceedings of the 8th Workshop on Vision, Modeling, and Visualization (VMV 2003), Munich, Germany, 2003.

[2] Blogpost: The mind of Conkerjo, *Spatial hashing implementation for fast 2D collisions*, available online: https://conkerjo.wordpress.com/2009/06/13/spatial-hashing-implementation-for-fast-2d-collisions/

[3] Ramon Santamaria (@raysan5), raylib: a simple and easy-to-use library to enjoy videogames programming, available online: https://www.raylib.com/