University of Tromsø

INF-2200

Assignment 4 - Bouncing Balls

Alexander Einshøj

Department of Computer Science

November 20, 2015

1 Introduction

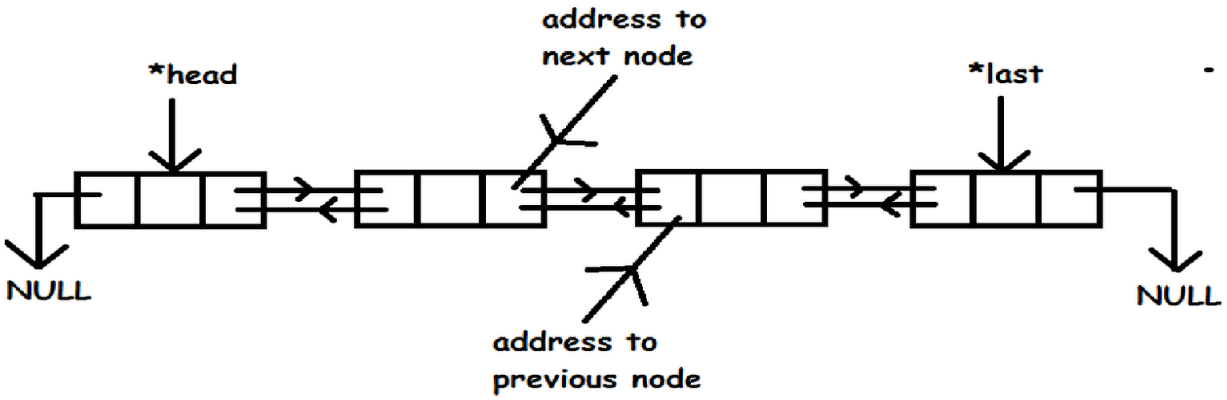
In this paper, a variant of the classic bouncing ball animation is implemented and explained. The main task explained in this paper, is the implementation of linked lists, and its uses. source:

2 Technical Background

2.1 Linked list

A linked list is a structure used for keeping track of several elements, with the opportunity to iterate through the list. The list consists of nodes, which each define a pointer to the next node in the list, and the data of the respective node. In this implementation, a doubly linked list is implemented instead of a singly linked list. In a doubly linked list, each node will consist of a pointer to the previous node, a pointer to the element or data, and a pointer to the next node in the list. This creates the possibility of accessing the list from either the first or the last node, and to iterate in both directions. In the case of both kind of linked lists, the last node will point to NULL.

With doubly linked lists, the head will point to NULL in the other direction, when iterating backwards.



The above figure shows a graphic representation of a doubly linked list, and where each node points to in both directions.

2.2 SDL

SDL(Simple DirectMedia) is a library designed to provide access to elements such as keyboard, audio, and graphics hardware via OpenGL.
In this implementation, it's mainly used to provide the screen and graphics for the visual elements seen when running the program.

3 Design

This solution uses a doubly linked list, instead of the suggested linked list. Some functions from the precode are replaced by a more preferable implementation. The doubly linked list allows for easier further improvement to the code than a singly linked list could do, by being more flexible and by supporting more possibilities.

The list implementation is copied from the first INF-1101 home exam from spring 2015, and further edited by the author to fit in with the current design.

Each frame of the program is basically just a loop which starts with the screen being cleared. Following this, another loop adds more elements to the list if there is room, followed by the drawing and moving of the objects on screen. The fps is currently set at the beginning of the loop, and it should be rather high before the movements on the screen look smooth.

3.1 Implementation

The assignment is written in C.

4 Results & Discussion

The implementation as is, fulfills all given requirements. All the objects bounce, they rotate, and they are replaced after a given amount or time, or if they move too slow. Gravity is simplified, but implemented. The solution is currently set to produce a list of up to 50 elements, with random starting speed in both directions.

When it comes to removing objects from the list, the function works as it should, but it is not called upon entirely correct. When deciding which ball to replace, the implementation looks for a ball that is laying relatively still, but other than that it's randomly chosen by the loop when looping through the objects on the screen. This is because of me not using the countdown variable from the precode, but trying to solve the issue by using some SDL functions.

As to why I chose doubly linked list instead of any of the other alternatives(trees, singly linked list etc), it's because of the simplicity of implementation and flexibility for further development.

The solution seems to produce a heavy lag after running for a bit, but it seems as though this is pretty hard to avoid when using SDL this way, for some reason. I do not know why, as all the elements are freed from memory and also from screen.

5 Conclusion

Getting any results from an implementation like this is difficult, as there are not many measuring points. The program was run through valgrind to ensure no memory leak or errors, successfully. For this task specifically, a singly linked list may have been more convenient to implement and use, and the use of SDL functions will probably be avoided as much as possible by the author in future implementations, given this is the reason for the heavy lag.

Bibliography

Sdl, November 2015. URL `https://www.libsdl.org/`.

Steffen Valvåg. Home exam 1, 2301-datastrukturer og algoritmer, November 2015.