WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

SIMPLE API FOR 3D APPLICATIONS

Author:
Lucian F. Ştefănoaica

Scientific Coordinator:
Lector Dr. Marc E. Frîncu

Timişoara, 2015

Contents

Abstract

The application programming interface which is presented in this thesis, alongside implementation related concepts, is actually a small library packed with a couple of graphical effects based on particle systems. The thesis is composed out of four chapters and the content of each chapter is shortly described bellow.

In the *Introduction* chapter, a brief history of computer graphics and particle systems is given. This chapter also specifies the role that particle systems play in computer graphics and why are they necessary.

The *Particle Systems* chapter obviously describes some technical details about particle systems. By the end of this chapter a programmer should already have an idea about how to implement such a system in a graphical application.

The *Application* chapter presents all the particle based graphical effects in the API and gives implementation details about them. It also gives details about the structure of the API and that of the application which uses the API. Besides this it demonstrates the API's use with a couple of screen-shots.

Very few software pieces are perfect at their first implementation. The *Conclusion* chapter emphasizes some changes which can be made in order to improve the performance of the API and gives a short description of my learning experience.

Abstract

Abstract in Romanian.

Chapter 1

Introduction

1.1 Brief History of Computer Graphics

First of all, what is *computer graphics*? The term first appeared in 1960 and it was made-up by William Fetter who was, at the time, a computer graphics researcher for Boeing. The term refers to a subfield of computer science which deals with image data representation and manipulation using computers.

The first computer able to do graphics is the Whirlwind computer. Its development was started in 1945 at MIT by a team of computer scientists led by Jay Forrester. The purpose of this computer was to make aircraft tracking possible on a large oscilloscope screen via a graphical application. Although the aircraft tracking application was the first application in the field of computer graphics it was not interactive. It could only display the real time positions of the tracked aircrafts. The first interactive graphical application was Tennis For Two and it was created by William Higinbotham because he wanted to kill the boredom of the visitors of the Brookhaven National Laboratory.

In 1959 the TX-2 computer emerged. This computer was used by Ivan Sutherland to program the Sketchpad, a tool for creating very precise engineering drawings. The software offered its users the possibility to draw lines and circle arcs. The lines could then be made perfectly parallel or perpendicular in order to fit the users drawing needs. Sketchpad is known to be the first graphical user interface or GUI in short form.

In 1966 Ivan Sutherland made yet another contribution to the field of

computer graphics by inventing the Sword of Damocles the first computer controlled head mounted display. This device displayed two stereoscopic images of the same wire-frame mesh. Two decades later NASA would use his methods in virtual reality research.

Very soon after, in 1970 actually, the field of computer graphics was upgraded by Henri Gouraud, Jim Blinn and Bui Tuong Phong. The first added the Gouraud shading model and the other two added the Blinn-Phong shading model. In 1978 Jim Blinn also added bump mapping to the computer graphics field.

1.2 Brief History of Particle Systems

The term *particle system* was coined in 1982 by William T. Reeves who was a computer graphics researcher for Lucasfilm Ltd., a film production company known for films like the Star Trek and Star Wars franchises. But what is a particle system? Here's the original definition:

"A particle system is a collection of many minute particles that together represent a fuzzy object. Over a period of time, particles are generated into a system, move and change from within the system, and die from the system."

–William T. Reeves–acm Transactions On Graphics–April 1983–Vol. 2, No. 2

At Lucasfilm Ltd. Reeves helped to develop the wall of fire effect which occurred every time the Genesis Device was used in the film Star Trek II: The Wrath of Khan. This fictional device had the amazing ability to transform any lifeless planet into a perfectly habitable place. Though this is not the first time when particle systems have been used in the computer industry, this particular effect helped to coin down the term *particle systems*.

The first particle systems were implemented in the earliest video games where this concept was used to portray exploding spaceships which left behind many little dots or lines. For example "Spacewar!" which appeared in 1962 and "Asteroid" which appeared later in 1978 had such effects implemented. Since their dawn till the very present particle systems have been used to implement various interesting phenomena such as fire, smoke or waterfalls. In the absence of particle systems these phenomena would be very difficult to implement if not impossible.

1.3 The Necessity of Particle Systems

An alternate solution to the problem of modeling objects that are "fuzzy", as William T. Reeves used to call them, would be to use surface-based representations (the ones which are used to model objects like cars or spaceships). Now, if one starts to think a little it wouldn't be a very natural approach of the problem to try to model an object which has no smooth surfaces, perhaps a cloud of smoke, with a series of triangles where each triangle represents a surface which actually does not exist on the real object. Sure, one could easily implement such a smoke cloud using triangles to represent its margins but the results would not be very realistic. If a large number of very small triangles were to be used then the smoke cloud would look better, in fact the smaller the triangles and the larger the count of the triangles the better the quality of the image. But there is a downside to this approach.

Let's imagine what would happen to the previously mentioned smoke cloud if a sudden wind gust were to hit it. The smoke cloud would most certainly move in the direction of the wind. Only with the surface-based representation it would move in the direction of the wind as a whole just like a very large balloon would. The representation of the smoke cloud would not be very realistic because in real life when a smoke cloud gets hit by a lateral wind force some pieces of it go with the wind while others go in completely different directions or move slightly from their initial position. Those pieces of the smoke cloud are actually particles of smoke. So if particle systems are used to "implement" smoke clouds in real life, why wouldn't this concept be used as well in the implementation of a smoke cloud which would only exist inside a graphical application. Instead of representing only the surface of the smoke cloud using medium to large sized triangles one should represent the whole volume of the smoke cloud using many small sized triangles. This way, not only the visual aspects of the smoke cloud would be improved but also a great amount of flexibility would be added to it. This added flexibility will make the object behave realistically, in other words it will make the smoke cloud react like a real smoke cloud would react not like a balloon which is the case of the surface-based representation. This added flexibility represents the advantage of particle systems over surface-based representations.

1.4 Motivation

Particle systems represent a fairly important role in computer graphics because without them the problem of rendering natural phenomena like fire, smoke, waterfalls, fog and others would not be solved correctly. Sure there may be other techniques which can be used to treat this problem out there, but one should reckon that using particle systems is the most natural approach to this problem.

Programming a couple of particle systems represents a good way to learn how to program graphical applications. One of the reasons why I accepted to study particle systems and implement some graphical effects based on them is curiosity. I was curious of what stands behind those nice fiery graphical effects which appear in most computer games. So naturally I wanted to learn about them and what better way to learn than to implement my own. Another reason is because I wanted to help others learn to.

Typically most programmers use libraries. Why? Because it's faster. Most problems big or small can be divided into subproblems. If one divides a larger problem into a set of ten smaller subproblems he would then have to gradually solve each and every one of the subproblems in order to solve the larger problem. Let's say that every subproblem in the set takes about two to three days to solve, this means that the whole set of subproblems should take, in the worst case, about a month to solve. What if somebody else has already solved half of the subproblems in the set and has packaged the solutions in a library? Then that library would make any programmer who would face the initial larger problem twice as productive because that somebody else has already done half of the work. So another reason why I accepted to write the simple graphical API presented in this thesis is to help others by giving them a set of solutions to problems which they may encounter in writing graphical applications which are required to have particle based graphical effects.

Chapter 2

Particle Systems

2.1 What are particle systems?

This section presents all the specific details of particle systems.

Chapter 3

Application

This is the application chapter.

Chapter 4

Conclusion

This is the conclusion chapter.