Workshop Series: Smoke & Gas Particles

Benjamin Kenwright[1]*

Abstract
This practical focuses on smoke effects using a particle-based framework. The student needs to implement a real-time interactive smoke effect using Newtonian mechanics (e.g., collision detection, forces, and movement).

Keywords
Newtonian Mechanics, Particles, Constraints, Classical Mechanics

[1] *Workshop Series (www.xbdev.net) - Benjamin Kenwright*

Contents

Introduction

Smoke & Gas The topic of this practical is the implementation of a real-time particle smoke simulation (i.e., fluid dynamic motion of particle bodies in a complex scene). The user should be able to interact with the simulation while it is running (e.g., through the mouse or keyboard) to control the 'particles' - such as, adding forces so particles swish and swirl around. The 'particle system' should be implemented using classical mechanic principles (e.g., Euler integration, penalty forces for collisions, sphere-sphere, and sphere-plane collision detection for the ground and rigid objects).

Tasks

1. Visually display various unique particle emitter types (e.g., billboards with different textures, 3D models) with characteristics, such as, transparency and size, that change over the particles life-span
2. User input (e.g., mouse or keyboard) to control and move the particles around. Implement unique features, such as, particles that surround animated shapes (e.g., halo), smoke from an exhaust (e.g., smoke and flames), rain (e.g., including splash effect when the rain hits the ground or objects), 3D text made up of particles (e.g., particles are emitted in a 3D shape and when they go outside the bounding volume they dissipate)
3. Particle system should be under the influence of gravity and environmental effects (e.g., wind and user interaction) - e.g., bouncing off and flowing around complex objects in the scene (i.e., sphere-plane collision detection)
4. Implement five or more 'unique' particle effects (gas, sparks, fire, dust, leaves, clouds, rain, ...) - including particles that interact with the environment (e.g., particles that bounce and 'engulf' a 3D scene)

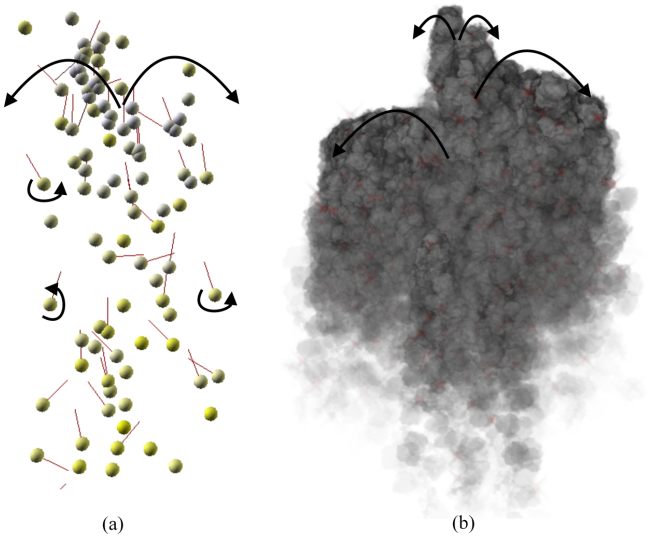

Figure 1. Demonstration Program - The practical program implements a bare-bone particle system (i.e., click the mouse on screen to place particle emitters); (a) simple point-mass particles with an ad-hoc spinning motion, and (b) a billboarded textured smoke effect.

1. Overview

Principles and Concepts A particle system enables us to create dynamic visual effects, such as, smoke, dust, sparks, and leaves. A particle system is a technique in computer graphics and video games that use a large number of very

small sprites or other graphic objects to simulate certain kinds of 'fuzzy' phenomena, which are otherwise very hard to reproduce with conventional rendering techniques - usually highly chaotic systems, natural phenomena, or processes caused by chemical reactions.

Creativity A vast assortment of effects can be created with particles. Exploiting technological advancement (e.g., GPU) and graphical techniques (e.g., height-maps), we are able to synthesize a banquet for the eyes - creating both beautiful and dynamic phenomena.

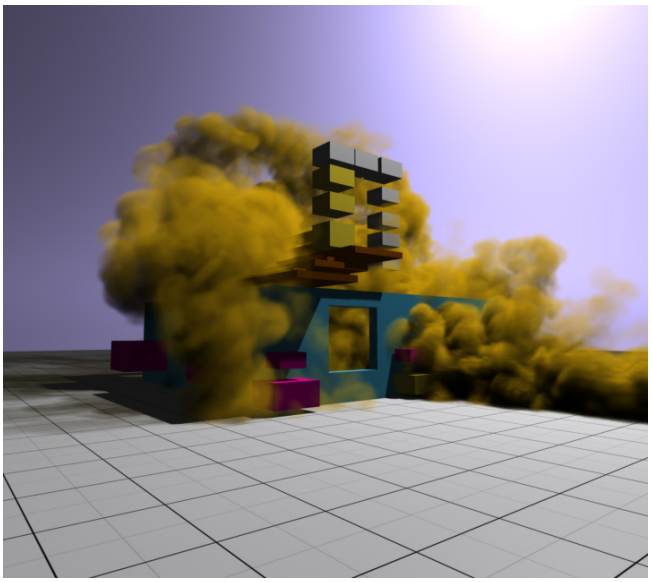

Figure 2. Animated Smoke Simulation - Real-Time interactive smoke simulation screen capture - showing particles dissipating in a scene - however, the particles are influenced by lighting and flow around objects in the environment.

2. Introduction

Billboarding A billboard within your game is always orientated to face the camera. As the camera moves the object is orientated to face the camera. The same theory can also be applied to cause an object to always face another object within your game. For example a head can be made to follow another object in your game.

Computational Speed A particle system may possess hundreds of thousands of particles. Hence, to ensure the system doesn't come to a grinding halt - we're able to incorporate numerical approximations into the implementation - since we're mostly interested in visual effect and not physical precision.

3. Implementation

Skeleton Starting with an uncomplicated particle-based implementation, the student should be able to expand the concept

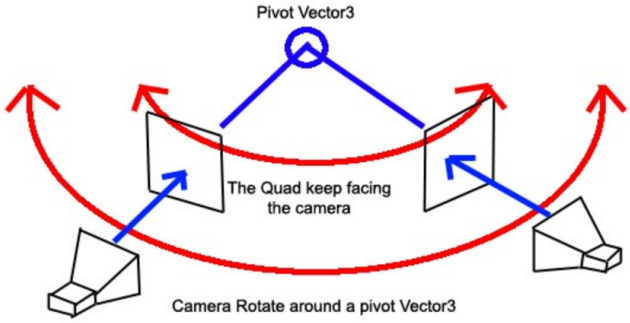

Figure 3. Billboarding - An visual trick to create the illusion of depth by having a quad always face towards the camera.

to include a vast assortment of visual effects. The student must be aware of both computational aspects and visual artifices (i.e., combination of different graphical and physics-based techniques).

Example Parameters A particle may possess an assortment of static and dynamic parameters. For example, some parameters you may include in your particle engine, may include:

- life-time
- colour (i.e., including alpha)
- size
- velocity
- mass
- acceleration
- orientation
- texture (or animated sprite frame number)

4. Summary

We have introduced graphical particle technique that can create a variety of free-flowing gas like effects (e.g., smoke, sparks, flames, leaves, dust). This particles are able to interact with the environment and the user to visually depict forces and movement within a scene.

5. Additional Exercises

This practical only gives a brief taste of smoke & gas particle principles. As an exercise for the student to help enhance their understanding:

Intermediate

- Implement an a scene with a vast number of particles (+million)
- Animate the particles (e.g., sprites or key-framed animation data)
- Add spherical 'magnetic' force-fields to a scene and dissipate particles and show how they are repelled or attracted to different regions as they progress through the environment
- Implement a GPU-based particle system

- Implement a scriptable particle system that can be used by an artist (i.e., save/create custom particle effects)

Acknowledgements

We would like to thank all the students for taking time out of their busy schedules to provide valuable and constructive feedback to make this practical more concise, informative, and correct. However, we would be pleased to hear your views on the following:

- Is the practical clear to follow?
- Are the examples and tasks achievable?
- Do you understand the objects?
- Did we missed anything?
- Any surprises?

The practicals provide a basic introduction for getting started with physics-based animation effects. So if you can provide any advice, tips, or hints during from your own exploration of simulation development, that you think would be indispensable for a student's learning and understanding, please don't hesitate to contact us so that we can make amendments and incorporate them into future practicals.

Recommended Reading

Physics for Game Developers, David M Bourg, Publisher: O'Reilly Media, ISBN: 978-1449392512

Computer Animation: Algorithms & Techniques, Rick Parent, Publisher: Morgan Kaufmann, ISBN: 978-0124158429

Code Complete: A Practical Handbook of Software Construction, Steve McConnell, ISBN: 978-0735619678

Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, ISBN: 978-0132350884

Game Inverse Kinematics: A Practical Introduction (2nd Edition) Kenwright. ISBN: 979-8670628204

Kinematics and Dynamics Paperback. Kenwright. ISBN: 978-1539595496

Game Collision Detection: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1511964104

Game C++ Programming: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1516838165

Computational Game Dynamics: Principles and Practice (Paperback). Kenwright. ISBN: 978-1501018398

Game Physics: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1471033971

Game Animation Techniques: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1523210688