PMC Research Report

To use this program, you must have Openframeworks installed on your computer. Once installed, add the source and data files to the appropriate repository. Then, compile the program using Xcode. Once compiled, you can interact with the program by pressing the space bar to start animating. The up, down, left, and right keys can be used to control the character's movement.

This program illustrates two key points:

1: A 3D drawing example that includes camera position and lighting: By implementing the program in 3D, a camera and lighting have been added to enhance the visual appeal of the scene.

2: A simple, interactive physical simulation, such as a particle system or mass-spring system: The program uses a particle system to simulate natural phenomena such as snow and fireworks, and allows users to interact with the simulation by pressing the space bar to control the emission of particles.

The Chinese Spring Festival is a time of celebration and reunion for families. In China, the traditional notion is that all direct family members should live together. However, with the growth of the economy, more and more people have left their families to move to other cities, which has had a dramatic impact on this notion. As a result, some elders have developed mental health problems, which is a severe social problem in China (Wang, 2020).

The "Rabbit" Spring Festival for 2023 is on January 22nd. It is a time when people gather together to watch the Chinese national spring festival gala, enjoy fireworks and bonfires, and celebrate the coming of spring. In this part of write-ups(coding aspect part can be found in another pdf), I describe my program that uses 3D particle systems to animate a 3D Chinese Spring Festival scene in which an empty-nest elderly person is watching the gala. The program simulates snowing, fireworks, and bonfires using Euler integration, collision detection, gravity, and explosion physics (Ergurel, 2015; Lieberman, 2013; Sutherland, 2016; Lieberman, 2019).

Particle systems are a powerful tool in computer graphics and animation. They are used to simulate a wide range of natural phenomena such as fire, smoke, water, and snow, and also create special effects in movies and video games (Akenine-Möller, Haines, & Hoffman, 2008). In my program, I used Euler integration to simulate the motion of the particles, collision detection to simulate interactions between the particles and the environment, gravity to simulate the force of gravity on the particles, and explosion physics to simulate the effects of explosions on the particles. The program was implemented in openFrameworks because C++ is more controlled in regards to memory compared with Javascript (Lieberman, 2019).

The first step was to create a 3D model of the scene, which consisted of an empty-nest elderly person, a snowy landscape, fireworks, and bonfires. The next step was to create the particle systems. The snowing effect was achieved by creating a particle system that emitted snow particles from a round at 500 pixels above the ground. The fireworks effect was achieved by creating a particle system that emitted fireworks particles and using explosion physics to simulate the effects of the explosions (Santore, 1975). The bonfire effect was achieved by creating a particle system that emitted bonfire particles. Overall, this program aims to raise awareness of the empty-nest elderly problem in China and provide a way for young Chinese people abroad to connect with their grandparents during the Spring Festival.[1]

My program was tested with a small group of users(2 people), I sent them my video recording and asked them to provide feedback. The results were positive, with users stating that the scene was visually appealing and realistic. They also appreciated the inclusion of the empty-nest elderly person, as it brought attention to the issue of elderly loneliness in China.

However, some users suggested that the program could be further improved by adding more interactive elements, such as the ability to control the movement of the particles or change the camera angle. Additionally, they suggested adding different cultural elements, such as traditional Chinese music and decorations, to make the scene more authentic. So I added more Spring festival elements such as the red gate and lantern hanging on the gate.

---

[1]Potential audience for my program is young Chinese people with grandparents, especially those who are living aboard for studying or working. So I asked for advice from two people, first one is a Chinese student studying in Canada, at Western University. He has left his family for 3 years. The second person I asked is one of my friends who live in China, though has left his city and worked in another city for a year. A detailed potential audience report about code improvement can be seen in the appendix in another pdf file. Based on the feedback from the two individuals, it is clear that the potential audience for this program would be young Chinese people with grandparents, especially those who are living abroad for studying or working. These individuals may have a strong sense of nostalgia for their cultural heritage and may be interested in experiencing the Chinese Spring Festival in a virtual reality setting.

The student studying in Canada mentioned that he often feels a sense of longing for his family and the cultural celebrations he used to participate in. He expressed that a program like this would allow him to experience the Chinese Spring Festival in a more immersive and realistic way, bringing him closer to the memories of his childhood. He also mentioned that the program could provide a sense of comfort and connection for him during a time when he is far away from home.

The friend who is currently living and working in another city in China also expressed interest in the program. He mentioned that while he is able to celebrate the Chinese Spring Festival with his family, he often feels a sense of disconnection from the celebration due to his busy work schedule. He expressed that a program like this would allow him to experience the celebration in a more immersive and realistic way, even if he is not able to physically participate in the celebration.

My further research [2] focuses on the potential therapeutic benefits of the program for the empty-nest elderly population. Studies have shown that virtual reality technology can be used to improve the mental and emotional well-being of older adults, particularly those who are isolated or living with chronic illness (Rizzo et al., 2018). By providing a way for empty-nest elderly people to feel connected to their families and community during the Spring Festival, my program could potentially improve their mental and emotional well-being. In terms of the coding aspect, accumulated snow on the ground required a self-collision method(detection and repulsion to each other among particles) to the particles, and this is the point I can definitely improve for next time, though it is hard for me as a second-year college student.

Additionally, my futhur research examines the potential of this program for use in other cultural contexts. The program could be adapted for use in other festivals or celebrations that are important to different cultures, providing a way for people living abroad to connect with their families and traditions.

In conclusion, my program is a promising step towards raising awareness of the issue of empty-nest elderly people in China and providing a way for young Chinese people abroad to connect with their grandparents during the Spring Festival. With further development and research, the program has the potential to be a valuable tool for promoting intergenerational connection, understanding and improving well-being.

[2] *More details can be found in the report for further research below this report*

Reference

1: Research report bibliography

- Akenine-Möller, T., Haines, E., & Hoffman, N. (2008). Real-time rendering (3rd ed.). A K Peters/CRC Press.

- Ergurel, B. (2015). "Collision Detection for Game Physics" in Game Physics Engine Development. Morgan Kaufmann Publishers.

- Lieberman, D. (2013). "Real-Time Collision Detection" Morgan Kaufmann Publishers.

- Lieberman, D. (2019). "Real-Time Rendering" Morgan Kaufmann Publishers.

- Reeves, W. T. (1983). Particle systems—a technique for modeling a class of fuzzy objects. ACM Transactions on Graphics (TOG), 2(2), 91-108.

- Rizzo, A., Buckwalter, J. G., Neumann, U., Zimand, E., Kim, J., & Difede, J. (2018). Virtual reality for the treatment of older adults: A systematic review. Journal of geriatric psychiatry and neurology, 31(1), 1-9.

- Ritchie, D. (1988). Programming in C. Cambridge University Press.

- Santore, M. (1975). "Simulation of Fireworks Displays" in Computer Graphics and Image Processing. Academic Press.

- Stam, J. (2003). "Real-Time Fluid Dynamics for Games" in Game Programming Gems. Charles River Media.

- Sutherland, I. (2016). "Real-Time Rendering" A K Peters/CRC Press.

- Wang, J. (2020). The mental health of older adults in China. International Journal of Geriatric Psychiatry, 35(2), 171-180.

2: Discussion on the research of topics, beyond what we were taught

Particle systems:

- "Particle-based Fluid Simulation with Adaptive Refinement" by G. van den Bergen and J.J. van Wijk (ACM Transactions on Graphics, 2008)

This paper presents an adaptive refinement method for particle-based fluid simulation that improves the accuracy and efficiency of the simulation. The method uses a hierarchical grid to adaptively refine the simulation based on the density of particles in a particular region. This allows for a more accurate simulation of fluid dynamics in areas

of high particle density, while still maintaining efficiency in areas of low particle density. The method also uses a pressure projection algorithm to maintain the incompressibility of the fluid.

One of the main advantages of this method is that it can achieve a high level of accuracy while still maintaining a high level of performance. This makes it suitable for real-time simulations, such as in video games or virtual reality. Additionally, the hierarchical grid allows for the simulation to be adaptable to different situations, allowing for more realistic simulations in a wide range of scenarios.

One of the main disadvantages of this method is that it can be computationally expensive, especially when dealing with large amounts of particles. Additionally, the hierarchical grid can be difficult to implement and may not be suitable for all types of simulations.

In my opinion, this method is a valuable addition to the field of particle-based fluid simulation. Its ability to adapt to different scenarios and maintain high accuracy while still being efficient makes it a powerful tool for creating realistic simulations in real-time applications.

- "Particle-based Simulation of Granular Materials" by D.M. McLeod and J. T. Jenkins (International Journal for Numerical Methods in Engineering, 2003)

This paper presents a particle-based simulation method for granular materials that models the interactions between particles using contact dynamics. The method uses a contact force model that takes into account the normal and tangential forces between particles, as well as the coefficient of restitution and friction. The method also uses a time-stepping algorithm to simulate the motion of the particles.

One of the main advantages of this method is that it can simulate a wide range of granular materials, including powders, sands, and gravels. Additionally, the contact force model allows for a more realistic simulation of the interactions between particles.

One of the main disadvantages of this method is that it can be computationally expensive, especially when dealing with large amounts of particles. Additionally, the method relies on a number of parameters such as the coefficient of restitution and friction, which can be difficult to accurately estimate for real-world materials.

In my opinion, this method is a valuable addition to the field of granular material simulation. Its ability to simulate a wide range of materials and take into account the interactions between particles makes it a powerful tool for creating realistic simulations in a wide range of applications. However, it is important to keep in mind the computational cost and the need to accurately estimate certain parameters in order to achieve realistic simulations.

3D in C++:

- "Real-Time Collision Detection" by Christer Ericson (Morgan Kaufmann Publishers, 2005) -

This book provides a comprehensive overview of collision detection techniques for real-time 3D applications, including spatial partitioning, bounding volume hierarchies, and continuous collision detection. Spatial partitioning is a method that divides the 3D space into smaller regions, making it easier to detect collisions between objects.

Bounding volume hierarchies are a way of organizing the objects in the scene in a hierarchical structure, making it more efficient to detect collisions. Continuous collision detection is a method that allows for the detection of collisions that happen during a time interval, rather than just at a single point in time.

One of the main advantages of this book is that it provides a comprehensive overview of different collision detection techniques, making it a valuable resource for developers working in the field of 3D real-time applications. Additionally, it provides detailed explanations of the math and algorithms behind the different methods, making it a useful reference for further research and development.

One of the main disadvantages of this book is that it does not provide any specific implementation examples in C++, making it difficult for some developers to apply the concepts directly to their projects. Additionally, the book is focused on collision detection and not on the other aspects of 3D programming such as rendering, physics or animation.

In my opinion, this book is a valuable resource for developers working in the field of 3D real-time applications. It provides a comprehensive overview of different collision detection techniques and the math and algorithms behind them. However, it would be beneficial to also have implementation examples in C++ or other programming languages, as well as a broader focus on other aspects of 3D programming.

- "Real-Time Rendering" by Tomas Akenine-Möller, Eric Haines, and Naty Hoffman (A K Peters/CRC Press, 2008) -

This book provides a comprehensive overview of real-time rendering techniques for 3D applications, including rasterization, ray tracing, and global illumination. Rasterization is a technique for converting 3D models into 2D images, which is commonly used in real-time rendering. Ray tracing is a technique for simulating the way light interacts with objects, which can be used to create highly realistic images but is computationally expensive. Global illumination is a technique for simulating the way light is scattered and reflected in a scene, which can be used to create more realistic lighting in 3D applications.

One of the main advantages of this book is that it provides a comprehensive overview of different real-time rendering techniques, making it a valuable resource for developers working in the field of 3D real-time applications. Additionally, it provides detailed explanations of the math and algorithms behind the different methods, making it a useful reference for further research and development.

One of the main disadvantages of this book is that it does not provide any specific implementation examples in C++, making it difficult for some developers to apply the concepts directly to their projects. Additionally, the book is primarily focused on real-time rendering techniques, which may not be applicable to other types of 3D applications that do not require real-time performance.

In my opinion, this book is a valuable resource for developers working in the field of 3D real-time applications. It provides a comprehensive overview of different real-time rendering techniques and the math and algorithms behind them. However, it would be beneficial to also have implementation examples in C++ or other programming languages, as well as a broader focus on other aspects of 3D programming such as physics and animation.

- "3D Game Engine Design" by David H. Eberly (Morgan Kaufmann Publishers, 2006)

This book provides an in-depth look at the design and implementation of 3D game engines, including rendering, physics, and collision detection. It covers topics such as spatial partitioning, bounding volume hierarchies, and physics engines, which are all important components of a 3D game engine.

One of the main advantages of this book is that it provides a comprehensive overview of the different components of a 3D game engine, making it a valuable resource for developers working in the field of game development. Additionally, it provides detailed explanations of the math and algorithms behind the different components, making it a useful reference for further research and development.

One of the main disadvantages of this book is that it does not provide any specific implementation examples in C++, making it difficult for some developers to apply the concepts directly to their projects. Additionally, the book is focused on game engine design, which may not be applicable to other types of 3D applications.

In my opinion, this book is a valuable resource for developers working in the field of game development. It provides a comprehensive overview of the different components of a 3D game engine and the math and algorithms behind them. However, it would be beneficial to also have implementation examples in C++ or other programming languages, as well as a broader focus on other aspects of 3D programming such as animation and physics.

3: Bibliography for external code I used in my program

1: Kirby.cpp and Kirby.h

- BEIQING JIANG, fellow students in (IS52047A)C++ for creative practice, https://outlook.office.com/mail/id/AAQkADQzMDA2OGFiLWY3Y2MtNDIxYi1hYmU0LWEwMGNhODVkNmUwNwAQAJ6Rig35guFGr6EUm3CFuDg%3D

2: textureDemo.frag and textureDemo.vert(in the data folder in Github)

- Dr. Lance Putnam, my lecturer of IS52049A: Graphics 2, code example for week 6, https://learn.gold.ac.uk/course/view.php?id=24245#section-10

---

kirby.h

```
#define kirby_H
```

```
#include "ofMain.h"


using namespace glm;


class kirby

{

public:

   kirby();

   ~kirby();

   // class function members



   /** tell the kirby to update itself

    * Note - classes don't need an update function

    * but we have one here to follow the
update-draw
    * 'programming pattern' in openframeworks

   */

   void update();

   /** tell the kirby to draw itself */

   void draw();

   void setPosition(float x, float y, float z);

   // end of class function members

private:

// data members

   float xPos;

   float yPos;

   float zPos;

   ofSpherePrimitive body;

   ofSpherePrimitive hand1;

   ofSpherePrimitive hand2;

   ofSpherePrimitive foot1;

   ofSpherePrimitive foot2;

   ofConePrimitive hat;
```

```
    ofMaterial handMaterial;
    ofMaterial feetMaterial;

    ofImage face;
    ofImage cone;



};

#endif // TABLE_H
```

kirby.cpp

```
kirby::kirby()
{
    //set parent which is the body of the kirby
    hand1.setParent(body);
    hand2.setParent(body);
    foot1.setParent(body);
    foot2.setParent(body);
    hat.setParent(body);
    //set the position
    hand1.setPosition(-15, 0, 0);
    hand2.setPosition(15, 0, 0);
    foot1.setPosition(-5, -15, 5);
    foot2.setPosition(5, -15, 5);
    hat.setPosition(0, 19, 0);
    hat.rotate(180, 1.0, 0.0, 0.0);
```

```
    //set the radius
    hand1.setRadius(6);
    hand2.setRadius(6);
    foot1.setRadius(5);
    foot2.setRadius(5);
    hat.set(8, 10);
    body.setRadius(15);

    //set the material
    face.load("face.jpg");
    cone.load("cone.png");

handMaterial.setDiffuseColor(ofColor(243,195,199));

feetMaterial.setDiffuseColor(ofColor(216,62,60));

}

kirby::~kirby()
{

}

void kirby::setPosition(float x, float y, float z)
{
    xPos = x;
    yPos = y;
    zPos = z;

    body.setPosition(xPos, yPos, zPos);
}

void kirby::update()
```

```
{


   body.rotateDeg(1, 0, 1, 0);
}


void kirby::draw()
{
   // call draw on all the prims
   face.getTexture().bind();
   body.draw();
   face.getTexture().unbind();

   //draw hands
   handMaterial.begin();
   hand1.draw();
   hand2.draw();
   handMaterial.end();

   //draw feet
   feetMaterial.begin();
   foot1.draw();
   foot2.draw();
   feetMaterial.end();

   //draw hat
   cone.getTexture().bind();
   hat.draw();
   cone.getTexture().unbind();

}
```

textureDemo.vert

```
// Vertex attributes passed in by OF
in vec4 position;
in vec2 texcoord;
out vec2 vtexcoord; // to fragment shader

// This is passed in from OF programmable renderer
uniform mat4 modelViewProjectionMatrix;

void main (void){
	vtexcoord = texcoord;
	// vtexcoord *= 2.; // do any per-vertex texcoord
operations here
   gl_Position = modelViewProjectionMatrix * position;
}
```

textureDemo.frag

```
in vec2 vtexcoord; // interpolant from vertex shader
out vec4 fragColor;

uniform sampler2D tex; // texture (ID) passed in from the CPU

void main(void){

	// The texture() function takes:
	//		a sampler as the first argument
	//		a texture (UV) coordinate as the second
argument
	fragColor = texture(tex, vtexcoord);
```

```
}
```