THE BIG BANG

Final Report

Team 050

Amanda Reese
Sara Rue
Logan Maloy
Nick Candelaria
Levi Valdez

EXECUTIVE SUMMARY

Cars are dangerous. That's a given. They always have been, and they always will be. However, they are also our main source of transportation. Therefore, car companies are forced to spend thousands of dollars per vehicle testing their safety. Our project is intended to save those companies thousands with a accurate model (accurate by weight, but not by alloy metal type. For instance it is assumed that every car utilizes the same alloy metal.)

This project was created with the expectation that it would aid companies in testing vehicles by way of a 3D program. To accomplish this huge goal, our team turned to nVIDIA's Cg graphics code language, which utilizes the most advanced technology for rendering 3d images on nVIDIA's most technically advanced graphics cards.

The program ended up being a huge success with the ability to calculate damage on multiple barriers and vehicles. By first entering vehicle weight, vehicle size, and the same information for the barriers you create a virtual vehicle crash test site. The program prerenders the data, so we achieve maximum efficiency and speed when running this program.

Included in this program is the skin application, which allows the system to wrap six images of a vehicle around a polygonal object, to give the appearance of a real vehicle. The images are real pictures of the Front, Rear, Sides, Roof, and the Underside

of the two vehicles, which we have entered in this program. The 1995 GMC Vandura Van, and the 2002 GMC Sierra extended cab are the two vehicles that were chosen for this project. The images of the vehicle are then edited to remove the background and to make the lighting on the vehicles as close as possible.

This program does not utilize two or more processors, but instead uses a faster single processor, with the aid of two graphics processor. What is unique about this program is the fact that it utilizes the GNOME Interface, which prevents windows and other operating systems, except Mac OS X, from being able to mimic its interface.

INTRODUCTION

The title of our undertaking is “The Big Bang”. It involves virtual crash testing it also involves discovering the safest and most durable vehicle. Our project analyzes the physical information of various crash tests.

OUR PURPOSE

The purpose of our study was to create an accurate virtual model of a vehicle crash. Our simulation is extremely radical. We also wanted to use this program as a chance to learn new coding and programming techniques. We hope that this experience will help broaden our computer horizons.

THE SIGNIFICANCE OF OUR PROBLEM

What can our program do? Basically it tests the ability for a car to stand up in a head on collision with another automobile.

BACKGROUND INFORMATION

The general function of a car is transportation. When have you ever heard someone in this modern world say ‘I’ll just hop in my carriage and my wonderful Clydesdales will spring into action and I just might reach town by tomorrow!’? No, everyone at the present time says, ‘I’m going to jump in my car and head into town, I’ll be back in five minutes.’

In light of these facts, it makes perfect sense that we need to test these cars for safety. Each year, automobile companies spend thousands upon thousands of dollars to do exactly that. But we thought, ‘what if they could accurately test these cars, virtually?’

Therefore, we designed a basic program that would do exactly that. By factoring in the weight of the vehicle, assuming that they are made of the same basic alloy metals, and sending one car against the other, we have designed a virtual test crash site.

OUR GOAL

By the end of our project, we hope to achieve a new understanding of the intricacies of car crashes. We also hope that our program will operate with accuracy close to real-life car crashes, and perhaps be used to save the time and money of the larger companies.

MATERIALS

During the creation of our program, we incorporated the use of a Gateway Performance 500XL with one AMD Athlon XP 1700+ series processor running at 1500 Megahertz with 256 Megabytes of RAM. This system has a 40 Gigabyte hard drive, a NVIDA Geforce 4 MX 440 (running NV30 Emulation) and an ATI Radeon 7500. Running the Linux operating system: Red Hat Linux 8. Our coding included NVIDA Cg and Microsoft Visual C++.

We also used 1995 GMC Vandura Van Weight Specifications, and a 2002 GMC Sierra Extended Cab Truck Weight Specifications as the models preset in our program.

INFORMATION

Information About Cg

Cg was designed and developed by NVIDIA with the aid of Microsoft and is capable for rendering 3D objects faster that conventional methods. Cg sends the 3D code (OpenGL or DirectX) to the graphics processor first, instead of sending the instructions to the main processor.

Information About Kinetic Pulse

Kinetic Pulse is usually used in the study of bullet impacts and usually on armor plating. We used it for reasons of an object hitting an other object.

PROCEDURE

Initial step in developing the program is to gather information about all of the physical concepts involved in car collisions. Kinetic Pulse shows impact damage mathematically but with the aid of 3D programming we can render it graphically. Newtons Laws are the fundamental applications for motions included in reactions and recoil used in the program.

Our group decided to use OpenGL, which Cg (C for Graphics) utilizes, to render and compile the mathematical data into a car crash. OpenGL, is the easiest most common to work with in terms of 3 Dimensional programming.

Our program has not been included (Source code and Compiled Version) for the reasons; it still has OpenGL Syntax errors and the computer has to be running NV30 emulation (Which we will not be including in the html version April 10, 2003).

.

The program will display graphical representations of the collision and for that our team had to take 5 different photos of one single vehicle so the pictures could wrap around the polygon to resemble the vehicle.

RESULTS

Under 3D rendering the system utilized the formula for Kinetic Pulse and Newtons First Law. We were able to determin the impact and reaction of two vehicles coliding graphiclly but we have yet to find a way for OpenGL to print-out the mathematical data to the screen.

To view pictures goto

http://www.bsin.k12.nm.us/mesa/challenge/picts.html

MATHEMATICAL EQUATION

Kinetic Pulse: KP = (mv)(.5mv^2) m= Mass V= Speed (MPH)

In other words, if the VANDURA VAN was going to hit another identical van and both going at 50 Miles per hour either of the vans have a stoping power of 3.979357 x 10 ^ 15 pounds per square foot.

CONCLUSION

At this state we will with heavy hearts conclude our program. We hope people worldwide will someday turn to our programs like ours with high hopes that it will help to save lives. We have been told that this program can and may someday save lives. While the program was in the making it enriched our brains with superior knowledge. That way someday we can take over the World!!!

ACKNOWLEDGEMENTS

We would like to thank Jim (for work on math modeling) and John Reese (for information on GM class vehicles).

We would also like to thank:

James N. Hall (http://www.xmission.com/~fractil/math/kp.html) for the work on Kinetic Pulse.

TechTV (http://www.Techtv.com) for the research on vehicle safety.

Kinetic Pulse Program Code

```
// Start of Recovery.java in Linux GNU \ GUI
// Program Name: recovery.java
// team 050

// Java core packages:

import java.awt.*;
import java.awt.image.*;
import java.awt.print.*;
import java.awt.font.*;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.awt.color.*;
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
import java.text.DecimalFormat;
import java.applet.*;
import java.lang.*;

// Java extention packages:
import javax.swing.*;

public class recovery extends JFrame {
   private JDesktopPane oryxDesktop;

   // set up GUI
   public recovery()
   {
      super("Kinetic Pulse | 3D rendered version will be shown at Los Alamos");

      // create menu bar, menu and items
      JMenuBar bar = new JMenuBar();
      JMenu addMenu = new JMenu("File");
      JMenuItem newFrame = new JMenuItem("New Session");

      addMenu.add(newFrame);
      bar.add(addMenu);

      setJMenuBar(bar);

      // set up oryxDesktop
      oryxDesktop = new JDesktopPane();
```

```
    getContentPane().add(oryxDesktop);

    // set up listener for newFrame menu item
    newFrame.addActionListener(

    // anonymous inner class to handle menu item event
    new ActionListener() {

       // display new internal window
       public void actionPerformed(ActionEvent event) {

          // create internal frame
          JInternalFrame frame = new JInternalFrame(
          "Internal Frame", true, true, true, true);

          // attach panel to internal frame content pane
          Container container = frame.getContentPane();
          OryxStart panel = new OryxStart();
          container.add(panel, BorderLayout.CENTER);

          // set size internal frameto size of its contents
          frame.pack();

          // attaxh internal frame to desktop and show it
          oryxDesktop.add(frame);
          frame.setVisible(true);

       }

    } // end anonymous inner class

    ); //end call to addActionListener

    setSize(900, 700);
    setVisible(true);

} // end constructor

// execute application
public static void main(String args[])
{
    recovery application = new recovery();

    application.setDefaultCloseOperation(
    JFrame.EXIT_ON_CLOSE);
}
```

```
}  // end class PopSolve

// class to display text on a panel
class OryxStart extends JFrame
   implements ActionListener {

   private JTextField inputField1, inputField2, inputField3, inputField4, outputField;
   private int number1, number2, number3, number4;
   private double result;

   public OryxStart()
   {
      super("Kinetic Pulse");

      Container container = getContentPane();
      container.setLayout(new GridLayout(4, 2));

      // contstuct text fields

      container.add(
      new JLabel("Weight of first vehicle", SwingConstants.RIGHT));
      inputField1 = new JTextField(10);
      container.add(inputField1);

      container.add(
      new JLabel("Weight of second vehicle?", SwingConstants.RIGHT));
      inputField2 = new JTextField(10);
      container.add(inputField2);

      container.add(
      new JLabel("speed of first vehicle", SwingConstants.RIGHT));
      inputField3 = new JTextField(10);
      container.add(inputField3);


      container.add(
      new JLabel("speed of first vehicle", SwingConstants.RIGHT));
      inputField3 = new JTextField(10);
      container.add(inputField4);
      inputField4.addActionListener(this);

      container.add(
      new JLabel("Result", SwingConstants.RIGHT));
      outputField = new JTextField();
      container.add(outputField);
```

```
        setSize(800, 600);
        setVisible(true);

    }

    // process events
    public void actionPerformed(ActionEvent event)
    {
        DecimalFormat precision3 = new DecimalFormat("0");

        outputField.setText(""); // clears the output field


        try {
            number1 = Integer.parseInt(inputField1.getText());
            number2 = Integer.parseInt(inputField2.getText());
            number3 = Integer.parseInt(inputField3.getText());
            number4 = Integer.parseInt(inputField3.getText());

            result = ((number1 * number2) * (.5 * (number3 * number4)));
            outputField.setText(precision3.format(result));
        }

        // process improperly formatted input
        catch (NumberFormatException numberFormatException) {
            JOptionPane.showMessageDialog(this,
            "You must enter a valid integer",
            "Invalid number format",
            JOptionPane.ERROR_MESSAGE);
        }
    }

    // excecute third part
    public static void main(String args[])
    {
        recovery application = new recovery();

        application.setDefaultCloseOperation(
        JFrame.EXIT_ON_CLOSE);
    }

}
```