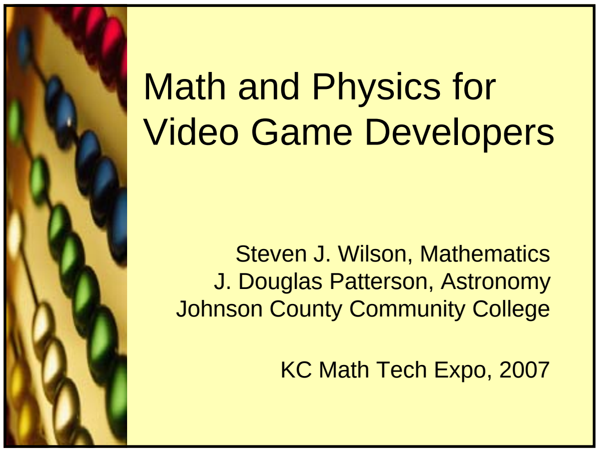
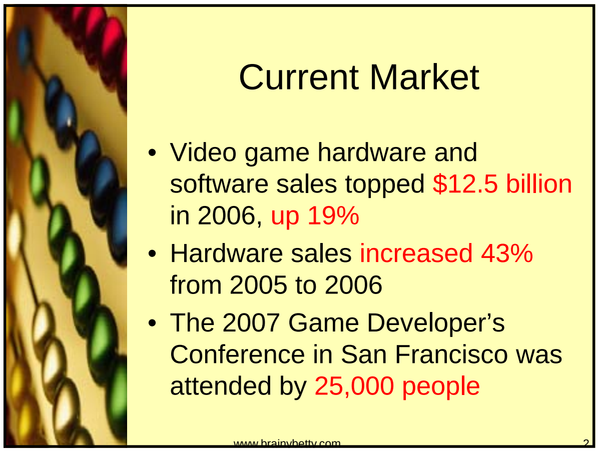


Math and Physics for Video Game Developers

Steven J. Wilson, Mathematics
J. Douglas Patterson, Astronomy
Johnson County Community College

KC Math Tech Expo, 2007

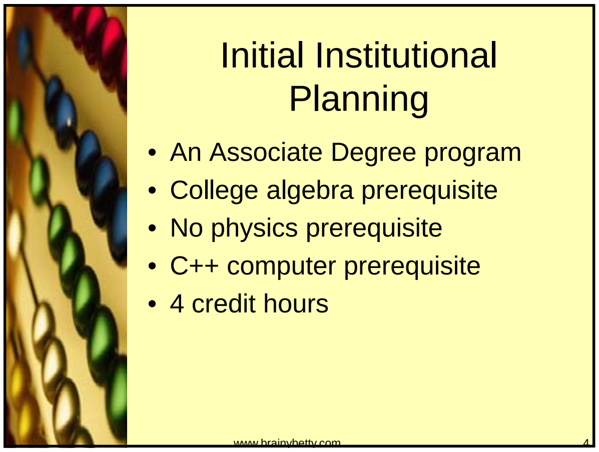


Current Market

- Video game hardware and software sales topped $12.5 billion in 2006, up 19%
- Hardware sales increased 43% from 2005 to 2006
- The 2007 Game Developer's Conference in San Francisco was attended by 25,000 people

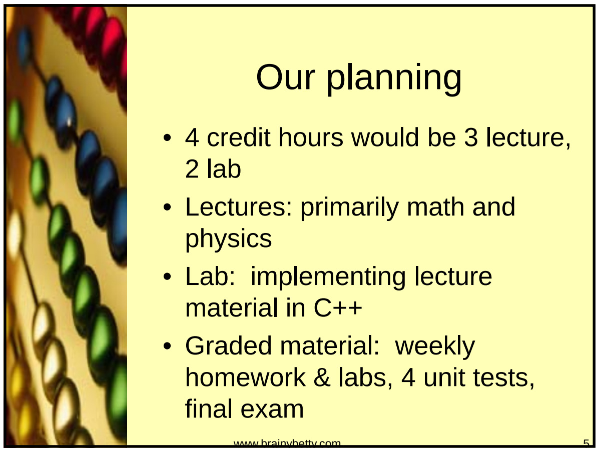
JCCC Interest

- Growth of Video Game market suggested a need for programmers
- Computer Science division was experiencing a shift in the programming job market



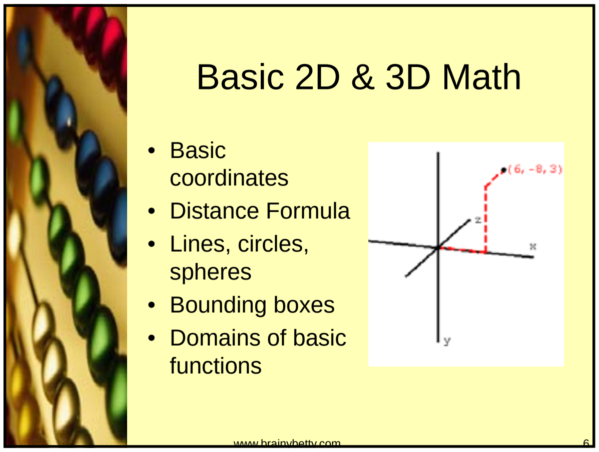
Initial Institutional Planning

- An Associate Degree program
- College algebra prerequisite
- No physics prerequisite
- C++ computer prerequisite
- 4 credit hours



Our planning

- 4 credit hours would be 3 lecture, 2 lab
- Lectures: primarily math and physics
- Lab: implementing lecture material in C++
- Graded material: weekly homework & labs, 4 unit tests, final exam



Basic 2D & 3D Math

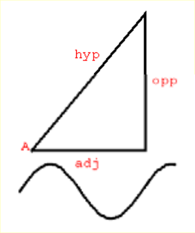
- Basic coordinates
- Distance Formula
- Lines, circles, spheres
- Bounding boxes
- Domains of basic functions

Basic Motion

- Units of measurement
- Position, velocity, acceleration in 1D
- 4 equations of motion (cst acceleration)
- Free Fall
- Forward Euler method

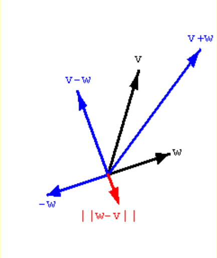
Basic Trigonometry

- Angle measures: degrees & radians
- Definitions & Graphs: sine, cosine
- Inverse functions: definitions, domain
- Identities: symmetric, translational
- Approximations: Taylor vs. Minimax



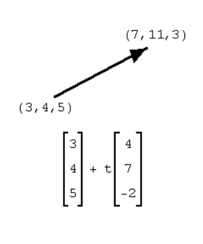
Basic Vectors

- Coordinate form
- Magnitude, heading and pitch
- Opposites, addition, subtraction
- Scaling, normalizing



Parametric Forms

- Line, Ray: defined by point and direction
- Convert to/from algebraic form
- Given point & direction, find new point
- Given 2 points, find direction

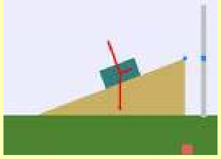


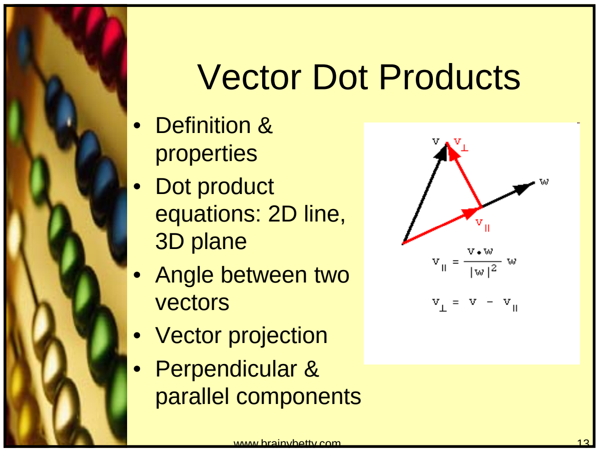
Projectile Motion

- Motion in 2D (generalizable to 3D)
- Vertical & horizontal components
- Acceleration due to gravity

Newton’s Laws

- Forces and accelerations
- Free body diagrams
- Normal forces
- Flat surfaces and Inclined planes
- Friction
- Wind resistance



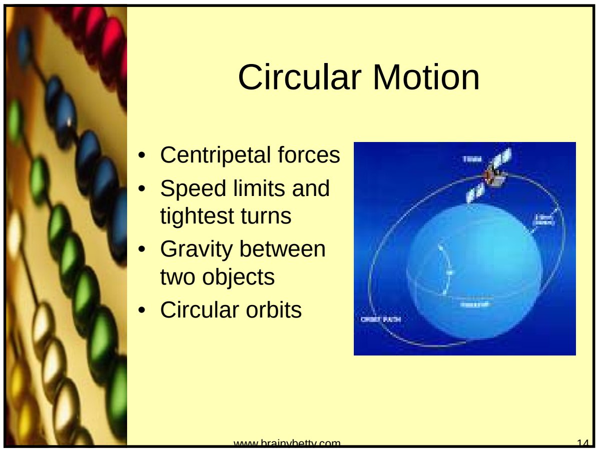


Vector Dot Products

- Definition & properties
- Dot product equations: 2D line, 3D plane
- Angle between two vectors
- Vector projection
- Perpendicular & parallel components

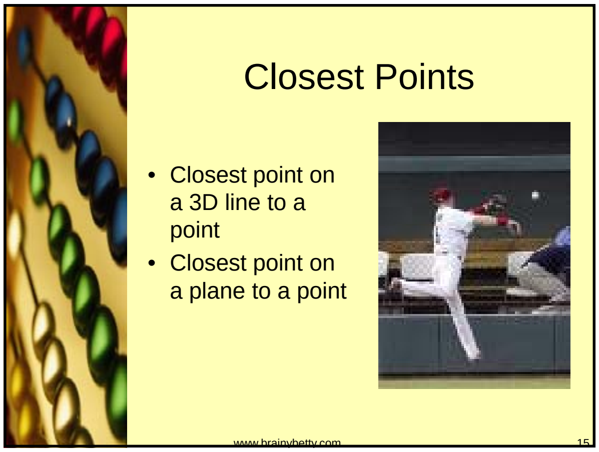
$$v_\parallel = \frac{v \cdot w}{|w|^2} w$$

$$v_\perp = v - v_\parallel$$



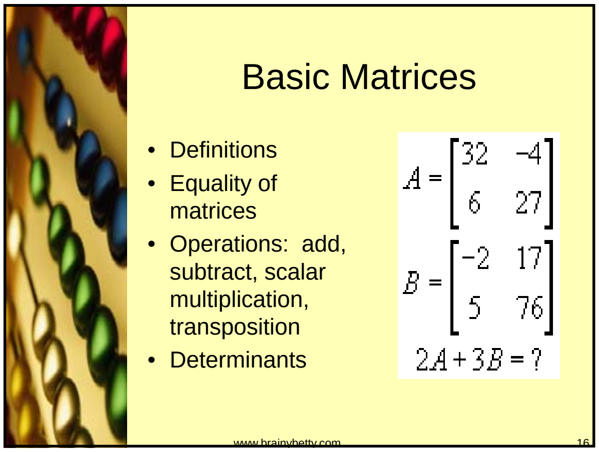
Circular Motion

- Centripetal forces
- Speed limits and tightest turns
- Gravity between two objects
- Circular orbits



Closest Points

- Closest point on a 3D line to a point
- Closest point on a plane to a point



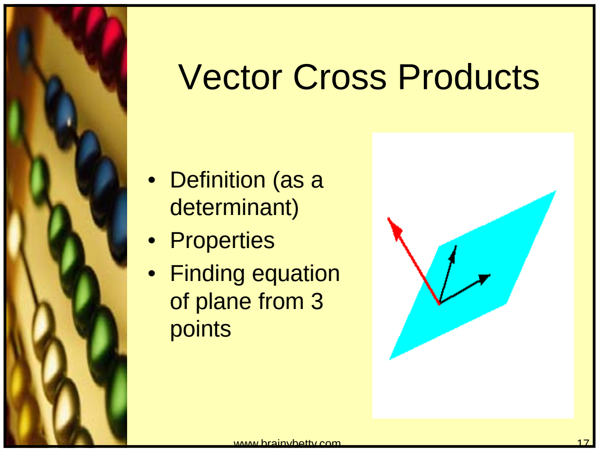
Basic Matrices

- Definitions
- Equality of matrices
- Operations: add, subtract, scalar multiplication, transposition
- Determinants

$$A = \begin{bmatrix} 32 & -4 \\ 6 & 27 \end{bmatrix}$$

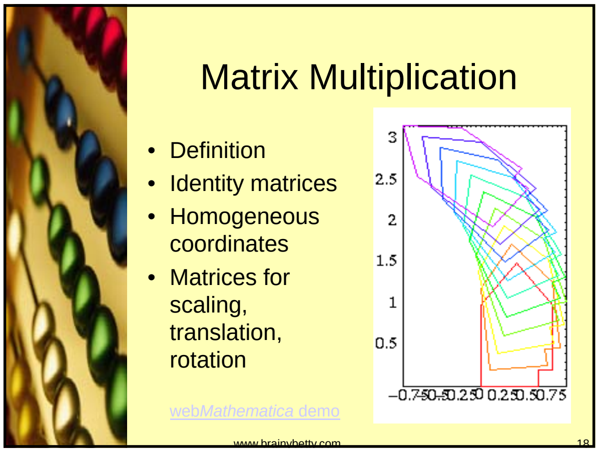
$$B = \begin{bmatrix} -2 & 17 \\ 5 & 76 \end{bmatrix}$$

$$2A + 3B = ?$$



Vector Cross Products

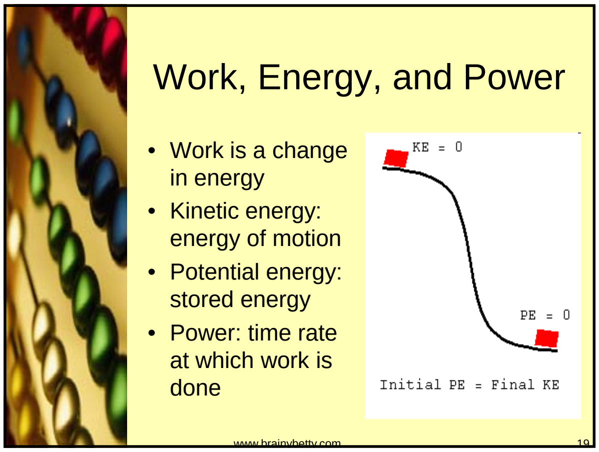
- Definition (as a determinant)
- Properties
- Finding equation of plane from 3 points



Matrix Multiplication

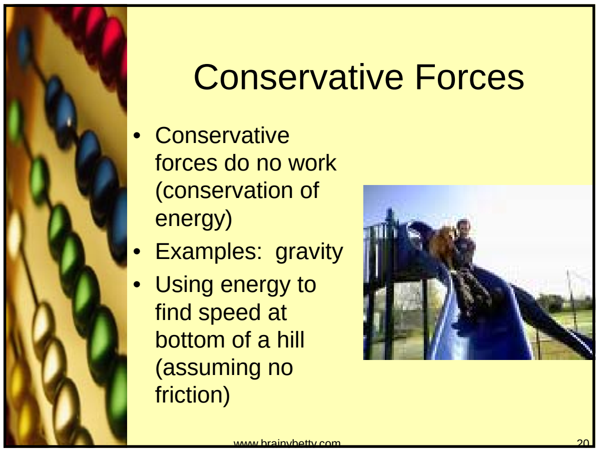
- Definition
- Identity matrices
- Homogeneous coordinates
- Matrices for scaling, translation, rotation

web*Mathematica* demo



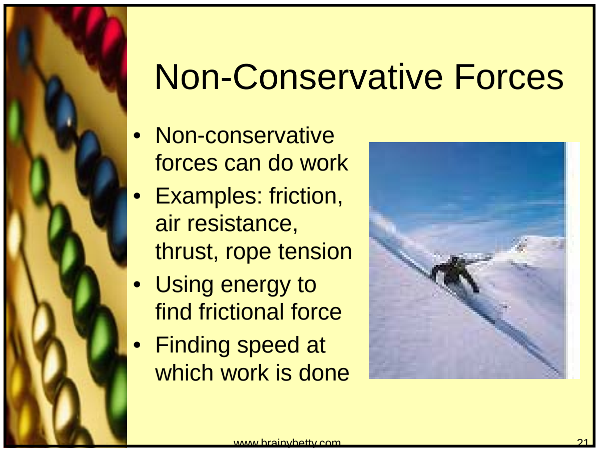
Work, Energy, and Power

- Work is a change in energy
- Kinetic energy: energy of motion
- Potential energy: stored energy
- Power: time rate at which work is done



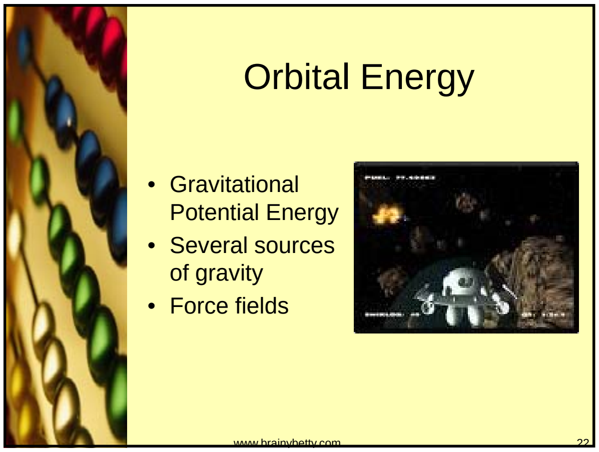
Conservative Forces

- Conservative forces do no work (conservation of energy)
- Examples: gravity
- Using energy to find speed at bottom of a hill (assuming no friction)



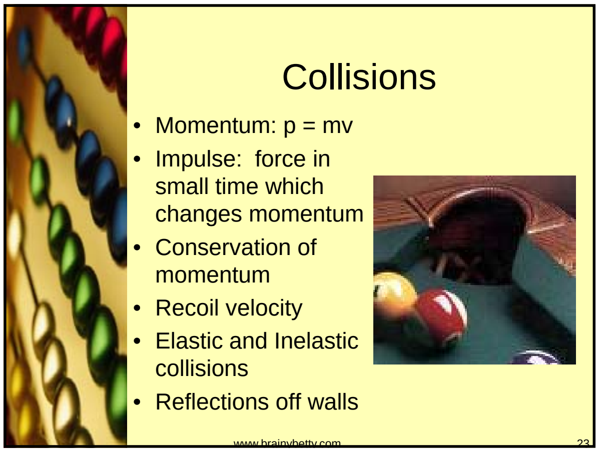
Non-Conservative Forces

- Non-conservative forces can do work
- Examples: friction, air resistance, thrust, rope tension
- Using energy to find frictional force
- Finding speed at which work is done



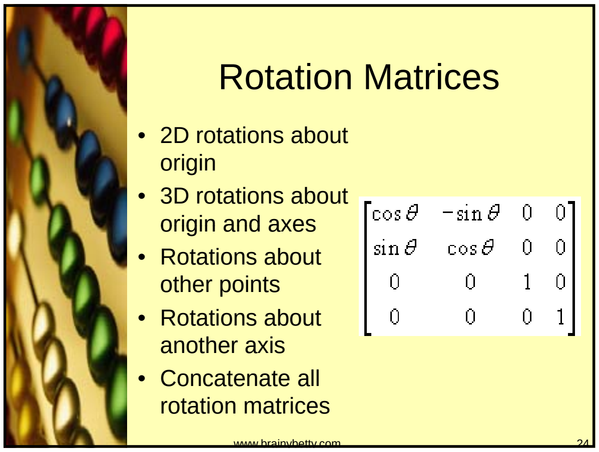
Orbital Energy

- Gravitational Potential Energy
- Several sources of gravity
- Force fields



Collisions

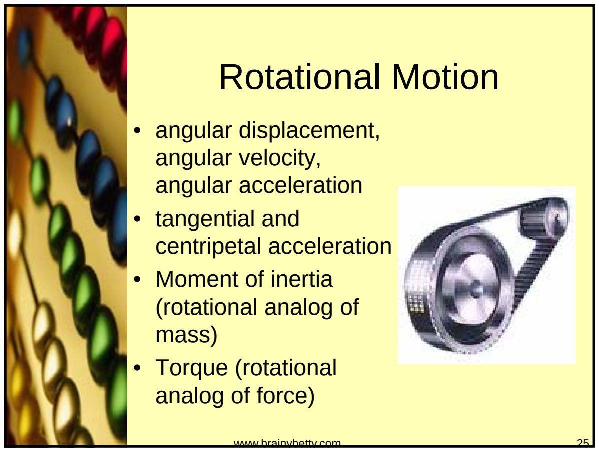
- Momentum: p = mv
- Impulse: force in small time which changes momentum
- Conservation of momentum
- Recoil velocity
- Elastic and Inelastic collisions
- Reflections off walls



Rotation Matrices

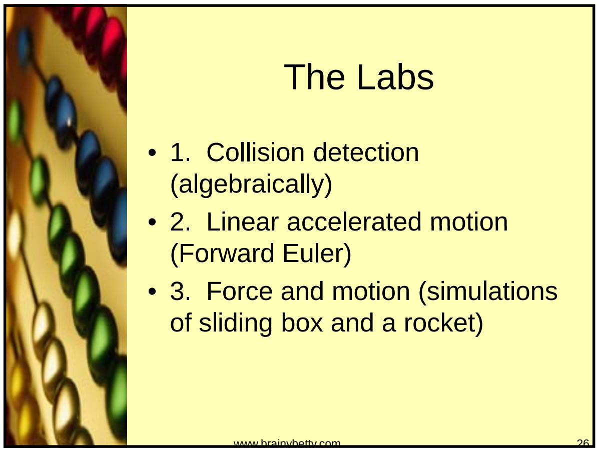
- 2D rotations about origin
- 3D rotations about origin and axes
- Rotations about other points
- Rotations about another axis
- Concatenate all rotation matrices

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



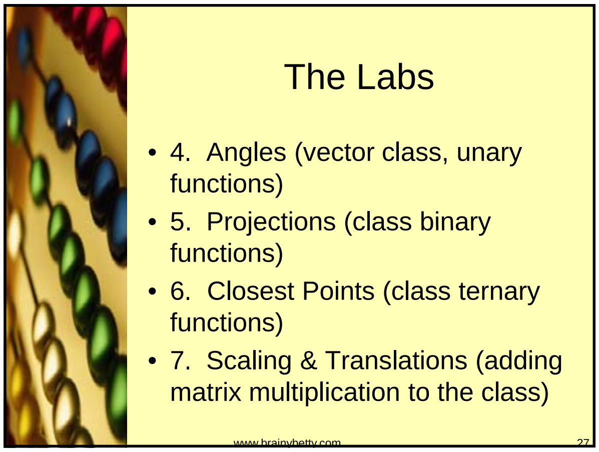
Rotational Motion

- angular displacement, angular velocity, angular acceleration
- tangential and centripetal acceleration
- Moment of inertia (rotational analog of mass)
- Torque (rotational analog of force)



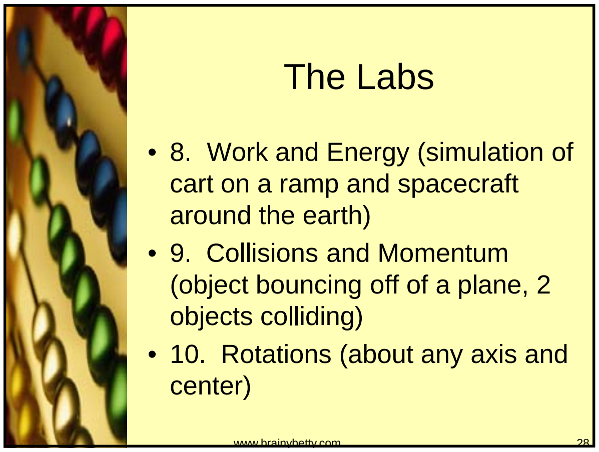
The Labs

- 1. Collision detection (algebraically)
- 2. Linear accelerated motion (Forward Euler)
- 3. Force and motion (simulations of sliding box and a rocket)



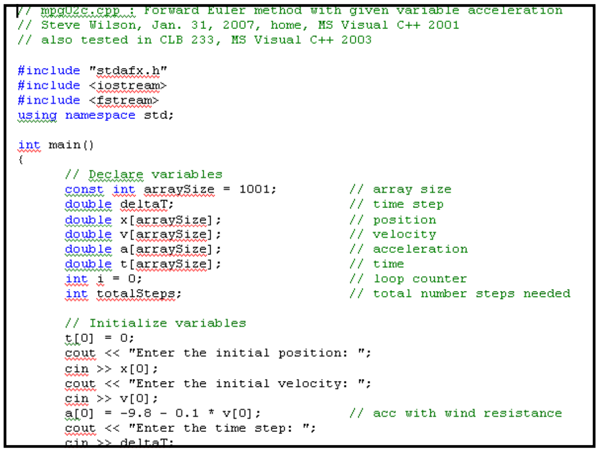
The Labs

- 4. Angles (vector class, unary functions)
- 5. Projections (class binary functions)
- 6. Closest Points (class ternary functions)
- 7. Scaling & Translations (adding matrix multiplication to the class)

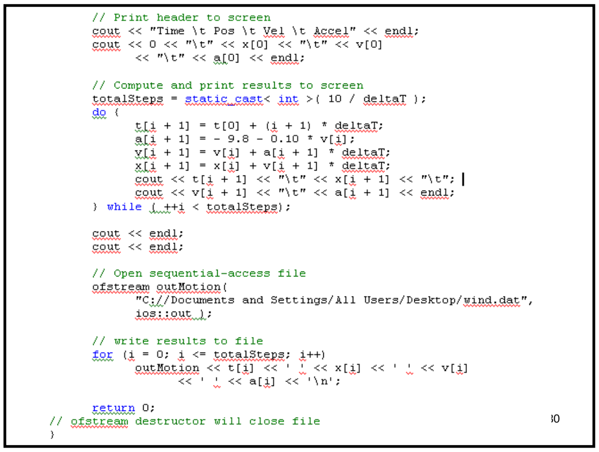


The Labs

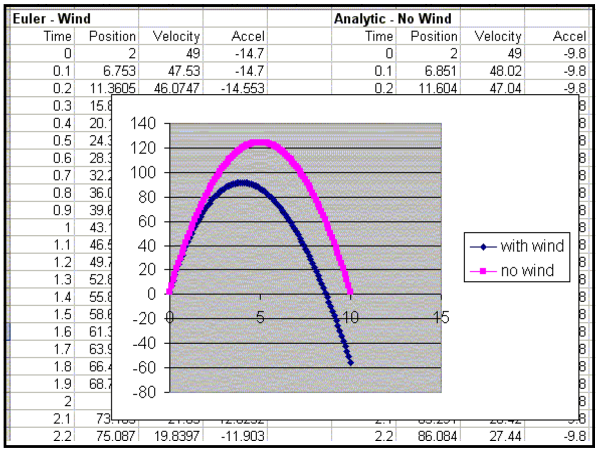
- 8. Work and Energy (simulation of cart on a ramp and spacecraft around the earth)
- 9. Collisions and Momentum (object bouncing off of a plane, 2 objects colliding)
- 10. Rotations (about any axis and center)



```
// mpg02c.cpp : Forward Euler method with given variable acceleration
// Steve Wilson, Jan. 31, 2007, home, MS Visual C++ 2001
// also tested in CLB 233, MS Visual C++ 2003

#include "stdafx.h"
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
        // Declare variables
        const int arraySize = 1001;                     // array size
        double deltaT;                                  // time step
        double x[arraySize];                            // position
        double v[arraySize];                            // velocity
        double a[arraySize];                            // acceleration
        double t[arraySize];                            // time
        int i = 0;                                      // loop counter
        int totalSteps;                                 // total number steps needed

        // Initialize variables
        t[0] = 0;
        cout << "Enter the initial position: ";
        cin >> x[0];
        cout << "Enter the initial velocity: ";
        cin >> v[0];
        a[0] = -9.8 - 0.1 * v[0];                       // acc with wind resistance
        cout << "Enter the time step: ";
        cin >> deltaT;
```



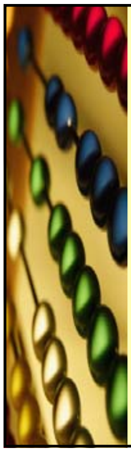
```
        // Print header to screen
        cout << "Time \t Pos \t Vel \t Accel" << endl;
        cout << 0 << "\t" << x[0] << "\t" << v[0]
                << "\t" << a[0] << endl;

        // Compute and print results to screen
        totalSteps = static_cast< int >( 10 / deltaT );
        do {
                t[i + 1] = t[0] + (i + 1) * deltaT;
                a[i + 1] = - 9.8 - 0.10 * v[i];
                v[i + 1] = v[i] + a[i + 1] * deltaT;
                x[i + 1] = x[i] + v[i + 1] * deltaT;
                cout << t[i + 1] << "\t" << x[i + 1] << "\t";
                cout << v[i + 1] << "\t" << a[i + 1] << endl;
        } while ( ++i < totalSteps);

        cout << endl;
        cout << endl;

        // Open sequential-access file
        ofstream outMotion(
                "C://Documents and Settings/All Users/Desktop/wind.dat",
                ios::out );

        // write results to file
        for (i = 0; i <= totalSteps; i++)
                outMotion << t[i] << ' ' << x[i] << ' ' << v[i]
                        << ' ' << a[i] << '\n';

        return 0;
// ofstream destructor will close file
}
```



Euler - Wind					Analytic - No Wind			
Time	Position	Velocity	Accel		Time	Position	Velocity	Accel
0	2	49	-14.7		0	2	49	-9.8
0.1	6.753	47.53	-14.7		0.1	6.851	48.02	-9.8
0.2	11.3605	46.0747	-14.553		0.2	11.604	47.04	-9.8
0.3	15.8							8
0.4	20.1							8
0.5	24.3							8
0.6	28.3							8
0.7	32.2							8
0.8	36.0							8
0.9	39.6							8
1	43.1							8
1.1	46.5							8
1.2	49.7							8
1.3	52.8							8
1.4	55.8							8
1.5	58.6							8
1.6	61.3							8
1.7	63.9							8
1.8	66.4							8
1.9	68.7							8
2								8
2.1	73	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	-9.8
2.2	75.087	19.8397	-11.903		2.2	86.084	27.44	-9.8

Students

- 8 students in Spring 2007
- 6 were Game degree candidates
- All had trig, 5 had some calculus
- C++ experience was varied
- Almost perfect attendance
- Grades: A, A, B, C, C, D, D, W
- Feedback: useful & challenging class



A Second Course ?

- Calculus prerequisite?
- Higher C++ prerequisite?
- Projections and viewing transformations
- Splines and Bezier Curves
- Quaternions
- Representing surfaces
- Lighting, shading, texturing