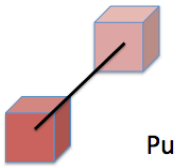
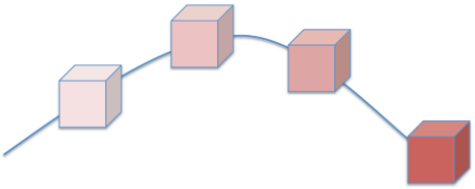
CSCE 4813 – Computer Graphics
Programming Project 2 – Due Friday 02/14/2020
Programming Project 3 – Due Monday 02/24/2020

1. Problem Statement:

The goal of this two-part programming project is to design and implement a video game called "angry bricks" modeled after the well known "angry birds" game. In our game, the player will grab onto a brick that is displayed on the lower left side of the screen with their mouse, drag it to the left, and when they release the mouse, the brick will "slingshot" towards a target on the right. As the brick moves across the screen, it will rotate in three dimensions, and fall towards the ground due the effects of gravity. When a brick hits the left or right wall it will "bounce" in the opposite direction. The brick will finally stop when it hits the ground. The goal of this game is to select a target, and see if you can slingshot the brick to that location.



Pull the brick back to left



Brick flies to the right when released

Project 2 – Creating and displaying the brick

Your first task is to create an OpenGL program that displays a colorful brick at a fixed location on the screen using the display callback. To do this, you will have to calculate the (x,y,z) locations of all 8 corners of the brick, and define the 6 faces of the brick using polygons with different colors.

Once you have this working, you can create a mouse callback function that captures the (x,y) location of the mouse, and redraws the colorful brick at that location. As you move your mouse around and click in different locations, the brick should be erased from the screen and redrawn in the new location. The key to this process will be converting from the (x,y) window coordinates into (x,y,z) object coordinates for display purposes.

Finally, you need to create a motion callback function that captures the (x,y) window coordinates of the mouse as you drag it across the screen. As the mouse moves, you should redraw the brick at the corresponding (x,y,z) location in object coordinates. For debugging purposes, you may want to print out the (x,y) window and (x,y,z) object coordinates as they are changing.

Project 3 – Simulating and displaying brick motion

In order to simulate brick motion you need to calculate the velocity vector from your "slingshot". To do this, you can record the (x1,y1) location where the mouse was clicked, and the (x2,y2) location where the mouse was released, and subtract these to get the direction and speed of the brick in object coordinates.

Your next task is to use the idle callback or the timer callback to "simulate physics" and update the (x,y,z) location of the brick at each time step and redisplay the brick. The tricky part here is to find a step size that results in "natural" looking motion. If the step size is too large, the brick will move too quickly. If the step size is too small, the brick will move in slow motion.

In order to simulate the brick "bounce" you will have to figure out when the brick hits the left or right wall, and update the brick position and velocity accordingly. To keep things simple, you can assume that the brick hits the wall squarely and bounces off perfectly in the opposite direction without change in speed and without introducing any strange brick rotations.

Finally, it would be nice to see the brick rotate slowly as it moves across the screen. To do this, you should keep track of the current location and rotation angles for the brick at each time step. Then you can use the glLoadIdentity, glRotatef, and glTranslatef functions to update the GL_MODELVIEW matrix prior to drawing the brick. This will make your brick rotate as it moves. Again, you need to select your rotation step size so it is not too fast and not too slow.

2. Design:

There are several design tasks you must complete for this project. First, you must design your brick model (a collection of 6 polygons should be enough). Next, you need to decide where to position the brick, and how to detect when the user has clicked on the brick. To do this, you will need to convert from "screen coordinates" to "object coordinates".

Your next task is to work out the equations needed to "simulate physics" so you can calculate the (x,y,z) locations of the brick as it is flying across the screen. If you ignore the effects of wind on the brick, it should follow a parabolic path. Finally, you need to figure out how you will detect when the brick hits the wall or the ground, and adjust the position and velocity accordingly.

3. Implementation:

This semester we will be using C++ and OpenGL to implement all of our programming projects. If you are using a Mac with Xcode installed, then you can download the src.tar file and compile the sample graphics code using the enclosed Makefile. If you are using a PC, then your best option would be to download and

install a Linux VM from the department's website. The instructions for doing this are posted in README file the "Source Code" page of the class website. Once you have Linux and OpenGL installed, you can compile your graphics program using "g++ -Wall project2.cpp -o project2-lGL -lGLU -lglut".

Using one of the programs in the source directory as a starting point, you can add code to the "display" function to display your brick on the screen. You will need to make use of the "mouse" callback to detect when the user has selected the brick, and the "motion" callback to track the motion of the mouse as the user drags it to the left, and the "idle" or "timer" function to update the location and orientation of the brick after it has been released by the slingshot towards the target.

Remember to use incremental development and good programming style when creating your program. Choose good names for variables and constants, use proper indenting for loops and conditionals, and include clear comments in your code. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save screen shots of your output in jpeg images for inclusion in your project report.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and all of your C++ program files. Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,

- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.