

## Computer Graphics 50:198:456/56:198:556 (Spring 2009)

<b>Homework:</b> 1	<b>Professor:</b> Suneeta Ramaswami
<b>Due Date:</b> 02/11/09	<b>E-mail:</b> rsuneeta@camden.rutgers.edu
<b>Office:</b> 321 BSB	<b>URL:</b> <a href="http://crab.rutgers.edu/~rsuneeta">http://crab.rutgers.edu/~rsuneeta</a>
	<b>Phone:</b> (856)-225-6439

---

### Programming Assignment #1

The goal of this programming assignment is to learn the basics of OpenGL and GLUT by implementing a simple game in which a *character* moves around in a region of the window called the *board*. The basic requirements are specified below. You are more than welcome to do more than asked! (Some suggestions are provided below.) Please note: Graduate students are required to implement at least one of the “optional” features below.

#### Program design:

First plan out your program design. The goal is to build a framework that allows you to add features to the game. While the simple game described here is quite small and may be implemented with a single `.cxx` file and no objects, it will be a useful exercise to use object-oriented design. For example, you might want to encapsulate a *character* class (which controls the motion and rendering of the moving character) and a *board* class (which stores the state of the environment in which the character is moving).

#### The character:

The character is the main source of interaction with the game. It is simply a shape that moves around on the board. You will need to keep track of its position for rendering and collision detection, and its speed and direction for maintaining its motion. The character must be drawn as an OpenGL object. Options include:

- A colored square (using `glRect*()`). This is the minimum requirement.
- A colored convex polygon (using `GL_POLYGON`). Use `glColor*()` to change the color of the character.
- A circle. This can be approximated as an  $n$ -sided polygon for some large  $n$  (20 or more).
- Any other 2D shape of your choosing (formed by multiple convex polygons, as per OpenGL’s requirements).

#### The board:

Create a board for the character to move around in. The character’s movement is confined to valid regions on the board. Options include:

- A simple border drawn around the window, inset a little from the window’s border.
- (Optional) Add rooms connected by doorways or other obstacles to the board. The character cannot pass through walls or obstacles.
- (Optional) Add doors to the outer borders of the board. When the character passes through this door, it wraps around to the other side of the board.

**Character movement:**

The character should be in constant motion around the board, depending on a speed and direction that is controlled by the user. At a minimum, the character should be able to move vertically and horizontally.

- Use `glutIdleFunc()` or `glutTimerFunc()` to create a smooth animation for the character.
- Use the arrow keys to control the movement direction. Use intuitive mappings, such as the `'↑'` for upward movement, `'→'` for rightward movement, and so on. See `glutSpecialFunc()`.
- Use the `'+'` and `'-'` keys to increase and decrease the character's speed.
- (Optional) Consider allowing the character to move in an arbitrary direction. The mouse is probably the best way to control the direction of motion.

**Collision Detection:**

The character should not be able to pass through the borders of the board. (One way to determine if the character's movement is legal is to check if the character's bounding box is within a valid region of the board. If the move is legal, only then update the character's position.) For example, if the character is moving left and hits a vertical wall, it stops there until the user changes direction.

**Other features:**

Use `'q'` or `'ESC'` to exit the game. The ASCII value of the `'ESC'` key is 27.

**What to turn in:**

Email me the **final version** of your program before midnight on February 11, 2009. Please include a README file explaining what your program does and how to interact with it. Include a Makefile if necessary. Bundle all required files together in one zip or tar file, and email it to me.