Assignment 1

- Simple video game

Paddle
Glut

- Graphics library utility toolkit.
- Provides windows, simple menus. (and removes the need for a lot of ugly code)
GlutCB Mouse
GlutCB Keyboard
GlutCB Menus

Main
Setup callbacks Glut
Hand control to OS

OS
The opengl pipeline (again)

Double buffering

Graphics App → Transform

- Translate
- Rotate
- Scale

→ Project

→ Clip

→ Rasterize

What? -- mostly polygons

BackBuffer

Blocking

Front Buffer
More glut

- `glutSwapBuffers()` – blocks the main program – holds program to a 60 Hz update rate
- `glutDisplayFunc()`
- `glutIdleFunc()` – always good for smooth animation
- `glutMotionFunc()`

- Demo all.
Drawing in OpenGL Graphics Primitives

- `glBegin( <feature type> );`
  - `GL_LINES, GL_TRIANGLES, GL_POINTS, GL_TRIANGLE_STRIP`
  - `GL_QUAD_STRIP, GL_TRIANGLE_FAN`

- `glColor();`
- `glVertex();`
- `glEnd();`

- Note there are also pixel-based operations that Bypass the frame buffer.

- Show an example using OpenGL first a simple triangle.
A circle, and a Sphere

- Circle class exercise. Use triangle Fan.
- Sphere review code.
Simple Animation for A1

- $x = x + dx; \; y = y + dy;$
- Bouncing off walls
- Gravity

- Must deal with contacts with paddle
- Steering the ball – increasing speed
- Must deal with contact with targets.

- Try to make a reasonable level of difficulty
Motion is the change in position with time \((dx, dy)\);

Acceleration is the second derivative of velocity in a vertical direction. \(Dy = dy - 0.01\)

Reflection on walls easy – not so easy for an oblique line. One method reflect and rotate.
Viewports (if time)

- Like windows within windows
- `glViewport(x, y, w, h)`
- Viewports are defined in terms of pixels.
- By default the viewport is the same as the glut window (it can be larger, it can be smaller).
- The Viewport

- The *scene* can be mapped into the viewport using
- `glOrtho(Sleft, Sright, Stop, Sbot, Snear, Sfar)`;
- The parameters are in scene coordinate.
Scene window to viewport mapping

- Like a window in the word

- Scene window to viewport mapping
Dealing with aspect ratios

- Three choices
  1. Let things distort but still fill the screen (default)
  2. Keep everything the same in terms of on screen sizes
  3. Keep scene window at a constant area with no distortion
Dealing with aspect ratios

```c
float scale;

winWid = w; winHeight = h;
scale = sqrt(winArea/(winWid*winHeight));
glViewport(0,0,w,h); // a sub area of the

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
// Define the scene window to viewport mapping
glOrtho(0.0,winWid*scale,0.0,winHeight*scale, -100.0, 100.0);
glMatrixMode(GL_MODELVIEW);
```