PGE input

- within PGE all non fixed objects are free moving
  - only circles (and springs) can be free moving at present
- we can interfere with circles by adding an impulse
- so we could
  - push it left with the left mouse button
  - push it right with the right mouse button
  - up with the middle mouse button

PGE input

-
  ```
  def mouse_hit (e):
      global m
      mouse = pge.pyg_to_unit_coord (e.pos)
      if e.button == 1:
          m.put_xvel (gb.get_xvel ()-0.3)
      elif e.button == 3:
          m.put_xvel (gb.get_xvel ()+0.3)
      elif gb.moving_towards (mouse[0], mouse[1]):
          pos = m.get_unit_coord ()
          # print ``mouse ='', mouse, ``ball ='', pos
          m.apply_impulse (pge.sub_coord (mouse, pos), 0.4)
      else:
          m.put_yvel (m.get_yvel ()+0.4)
  ```

PGE input

- in the main function we register the mouse event with our function

-
  ```
      pge.register_handler (mouse_hit, [MOUSEBUTTONDOWN])
  ```

- please see the implementation of breakout to see how this is integrated into a game breakout example ⟨`http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/pge/homepage.html`⟩

Collisions in PGE

- referring again to the breakout source code example ⟨`http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/pge/example_games.html`⟩

- notice that the section of code containing `delete_me` and `box_of`

Collisions in PGE

```
def delete_me (o, e):
    global blocks, winner, loser

    blocks.remove (o)
    o.rm ()
    if blocks == []:
        if not loser:
            winner = True
            pge.text (0.2, 0.3, ``Winner'', white, 100, 1)
            pge.at_time (4.0, finish_game)

def box_of (pos, width, height, color):
    global blocks

    blocks += [pge.box (pos[0], pos[1], width, height, color)\
       .fix ().on_collision (delete_me)]
```

Collisions in PGE

- the function `box_of` creates a blue box at `pos` with a `width` and `height`
- it also stipulates that this box is `fixed`
- furthermore if anything hit this box then the function `delete_me` is called

Collisions in PGE

- the function `delete_me` is a call back registered by the call to `on_collision` (described on the previous slide)

- this call back must be defined taking two parameters
  - the first, `o`, is the object whose callback is being called
  - the second, `e`, is the collision event which has describes the collision

- by using the event, `e`, it is possible to find out the other object in collision and other properties (if necessary)