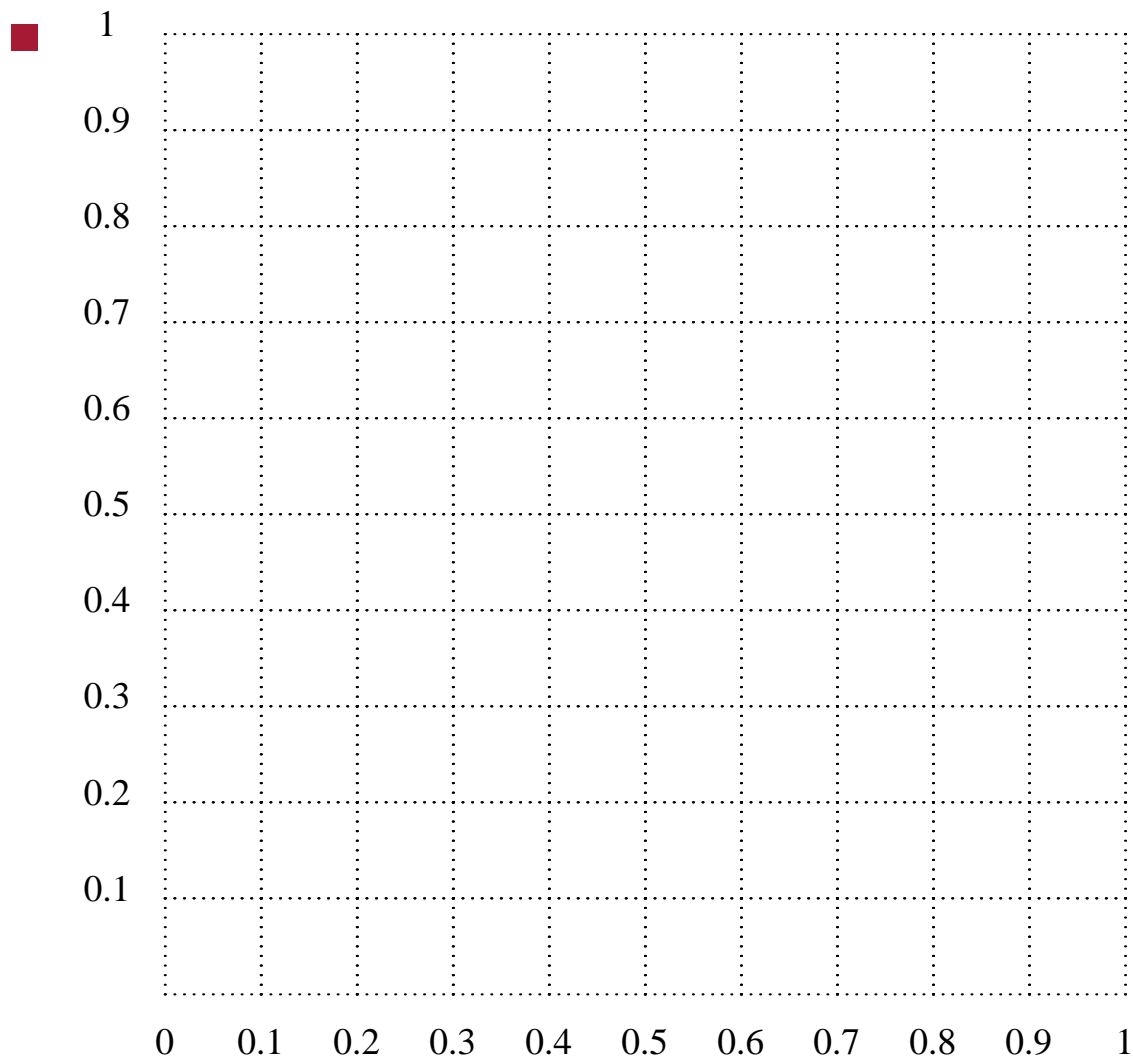


PGE Introduction

- PGE is a Predictive physics Game Engine
 - it operates by predicting the time of next collision rather than using a frame based approach

PGE Screen and world coordinates 1m x 1m



PGE works with Python and Pygame

- PGE allows you to create an object and give it physical attributes and pygame will draw the image
 - the object will behave as it would in the real world (almost!)
- you can interfere with the object using mouse and keyboard input
- your program can also modify the PGE world

PGE Python API

- is powerful enough to implement breakout in about 2 hours
- the Python interface allows you to:
 - define call backs to occur when objects collide
 - define timer functions to execute in the future
- you can also change an objects attribute (velocity/acceleration) in Python

PGE Objects

- colour
- circle
 - can be declared at any level
- polygon
 - can be declared at any level
- text
 - can be declared at any level other than level 0
- springs
 - can only be declared at level 0

PGE Objects

- only objects at level 0 are handled by the physics engine
 - objects declared at any level < 0 are in the background
 - objects declared at any level > 0 are in the foreground

- all objects are drawn in level order

- you **must** include a boarder to the PGE world
 - a picture frame to enclose all objects

- you **must** give a mass to non fixed objects

Simple example



tinypge.py

```
#!/usr/bin/env python

import pge, sys
from pygame.locals import *

print ``starting pge``
# pge.batch ()
pge.interactive ()

wood_light = pge.rgb (166.0/256.0, 124.0/256.0, 54.0/256.0)
wood_dark = pge.rgb (76.0/256.0, 47.0/256.0, 0.0)
red = pge.rgb (1.0, 0.0, 0.0)
ball_size = 0.05
boarder = 0.01
```

Simple example

tinypge.py

```
def myquit (e):
    print ``goodbye``
    sys.exit (0)

def key_pressed (e):
    if e.key == K_ESCAPE:
        myquit (e)

def placeBoarders (thickness, color):
    print ``placeBoarders``
    floor = pge.box (0.0, 0.0, 1.0, thickness, color).fix ()
    left = pge.box (0.0, 0.0, thickness, 1.0, color).fix ()
    right = pge.box (1.0-thickness, 0.0, thickness, 1.0, color).fix ()
    top = pge.box (0.0, 1.0-thickness, 1.0, thickness, color).fix ()
    return floor, left, right, top
```


Simple example

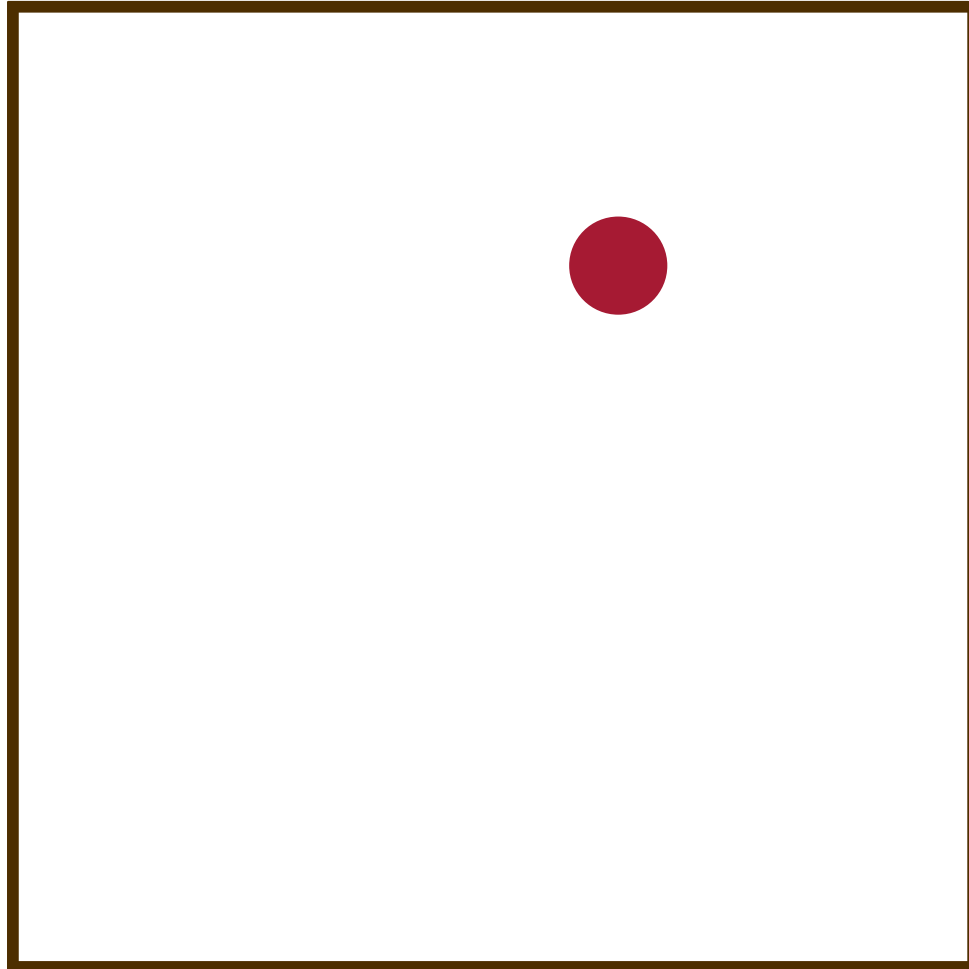
tinypge.py

```
def main ():
    pge.slow_down (6.0) # slows down real time by a factor of
    top, left, right, top = placeBoarders (boarder, wood_dark)
    pge.circle (0.6, 0.7, ball_size, red).mass (1.0)

    print ``before run``
    pge.gravity ()
    pge.draw_collision (True, False)
    pge.register_handler (myquit, [QUIT])
    pge.register_handler (key_pressed, [KEYDOWN])
    pge.display_set_mode ([800, 800])
    pge.run (10.0)
    pge.finish_record ()

print ``before main()``
main ()
```

Diagram for the world created by the simple code example



Simple example explanation

- PGE is run in interactive mode
 - it could be run in batch mode for testing
 - it can record a game, and the developer can play it back
 - single stepping each frame if necessary

- turn on gravity

- the circle is placed and given a mass

- `pge.draw_collision (True, False)`
 - draw an extra frame at the actual time of collision
 - do not colour vertices in collision

Simple example



tinypge.py

```
#!/usr/bin/env python

import pge, sys
from pygame.locals import *

print ``starting pge``
# pge.batch ()
pge.interactive ()

wood_light = pge.rgb (166.0/256.0, 124.0/256.0, 54.0/256.0)
wood_dark = pge.rgb (76.0/256.0, 47.0/256.0, 0.0)
red = pge.rgb (1.0, 0.0, 0.0)
ball_size = 0.05
boarder = 0.01
```

Simple example

tinypge.py

```
def myquit (e):
    print ``goodbye``
    sys.exit (0)

def key_pressed (e):
    if e.key == K_ESCAPE:
        myquit (e)

def placeBoarders (thickness, color):
    print ``placeBoarders``
    floor = pge.box (0.0, 0.0, 1.0, thickness, color).fix ()
    left = pge.box (0.0, 0.0, thickness, 1.0, color).fix ()
    right = pge.box (1.0-thickness, 0.0, thickness, 1.0, color).fix ()
    top = pge.box (0.0, 1.0-thickness, 1.0, thickness, color).fix ()
    return floor, left, right, top
```

Simple example

`tinypge.py`

```
def main ():
    pge.slow_down (6.0) # slows down real time by a factor of
    top, left, right, top = placeBoarders (boarder, wood_dark)
    pge.circle (0.6, 0.7, ball_size, red).mass (1.0)

    print ``before run``
    pge.gravity ()
    pge.draw_collision (True, False)
    pge.register_handler (myquit, [QUIT])
    pge.register_handler (key_pressed, [KEYDOWN])
    pge.display_set_mode ([800, 800])
    pge.run (10.0)
    pge.finish_record ()

print ``before main()``
main ()
```

Current limitations of PGE

- the screen must be square
 - xresolution must be the same as the yresolution
- only circles can be moving
 - springs act between a pair of objects (circles and/or polygons)

Features

- PGE can be asked to record the game
 - you can play it back again using the command line tool `pgeplayback`
 - useful if you want to debug your code

- it can generate frames at the time of impact, giving a greater sense of visual realism

- can slow down simulated time to make the game easier

Tutorial work and homework

- try out the examples in these notes and read the code
 - comment each line of code and read the [documentation](http://floppsie.comp.glam.ac.uk/Southwales/gaius/pge/homepage.html) to PGE aid your understanding

Balls bouncing ramps example



kongpge.py

```
#!/usr/bin/env python

import pge, sys
from pygame.locals import *

print ``starting kong in pge``
# pge.batch ()
pge.interactive ()
```

Balls bouncing ramps example



kongpge.py

```
t = pge.rgb (1.0/2.0, 2.0/3.0, 3.0/4.0)
wood_light = pge.rgb (166.0/256.0, 124.0/256.0, 54.0/256.0)
wood_dark = pge.rgb (76.0/256.0, 47.0/256.0, 0.0)
red = pge.rgb (1.0, 0.0, 0.0)
green = pge.rgb (0.0, 1.0, 0.0)
blue = pge.rgb (0.0, 0.0, 1.0)
blue_dark = pge.rgb (0.1, 0.1, 0.8)
steel = pge.rgb (0.5, 0.5, 0.5)
copper = pge.rgb (0.5, 0.3, 0.2)
gold = pge.rgb (0.8, 0.6, 0.15)
```

Balls bouncing ramps example



kongpge.py

```
ball_size = 0.02
boarder = 0.01
white = pge.rgb (1.0, 1.0, 1.0)
gap = 0.01
captured = None
sides = []
seconds_left = 120
previous = None
```

Balls bouncing ramps example



kongpge.py

```
def myquit (e):  
    print ``goodbye``  
    sys.exit (0)  
  
def key_pressed (e):  
    if e.key == K_ESCAPE:  
        myquit (e)  
  
def delete_ball (o, e):  
    for b in e.collision_between ():  
        if b != o:  
            b.rm ()  
            new_barrel ()
```

Balls bouncing ramps example



kongpge.py

```
def placeBoarders (thickness, color):  
    print ``placeBoarders``  
    floor = pge.box (0.0, 0.0, 1.0, thickness, color).  
        fix ().on_collision (delete_ball)  
    left = pge.box (0.0, 0.0, thickness, 1.0, color).  
        fix ()  
    right = pge.box (1.0-thickness, 0.0, thickness, 1.0, color).  
        fix ()  
    ceil = pge.box (0.0, 1.0-thickness, 1.0, thickness, color).  
        fix ()  
    return floor, left, right, ceil
```

Balls bouncing ramps example



kongpge.py

```
def placeRamps ():
    bot = pge.poly4 (0.95, 0.2,
                    0.15, 0.18,
                    0.15, 0.17,
                    0.95, 0.19, wood_dark).fix ()
    mid = pge.poly4 (0.85, 0.4,
                    0.05, 0.42,
                    0.05, 0.43,
                    0.85, 0.41, wood_dark).fix ()
    top = pge.poly4 (0.95, 0.7,
                    0.15, 0.68,
                    0.15, 0.67,
                    0.95, 0.69, wood_dark).fix ()
```

Balls bouncing ramps example



kongpge.py

```
def timer (e = None, param = None):
    global seconds_left, previous
    if seconds_left >= 0:
        pge.at_time (1.0, timer)
        s = ``%d`` % seconds_left
        if previous != None:
            previous.rm ()
        previous = pge.text (0.8, 0.9, s, white, 100, 1)
        seconds_left -= 1
```


Balls bouncing ramps example



`kongpge.py`

```
def new_barrel (e = None, param = None):  
    global gb  
    gb = pge.circle (0.9, 0.85, 0.03, steel).mass (1.25)
```

Balls bouncing ramps example

kongpge.py

```
def main ():
    global gb, sides

    placeRamps ()
    pge.slow_down (6.0) # slows down real time by a factor of
    timer ()
    new_barrel ()
    b1, b2, b3, b4 = placeBoarders (boarder, wood_dark)

    print ``before run``
    pge.gravity ()
    pge.draw_collision (True, False)
    pge.register_handler (myquit, [QUIT])
    pge.register_handler (key_pressed, [KEYDOWN])
    pge.display_set_mode ([800, 800])
    pge.run (seconds_left/6.0+4.0)
    pge.finish_record ()
```

Balls bouncing ramps example



kongpge.py

```
print ``before main()``  
main ()
```