Tutorial interpen.c

- writing test code is an important component of development
  - useful aid to debug code in development
  - useful regression test code
- `interpen.c` will be used by pge to help implement frame based collision (moving polygons)
- examine the file `pge/c/interpen.c` and the file `pge/c/interpen.h`
- the header file defines the exported functions and exported datatypes
- write 2 simple unit tests for each exported function
- compile and link your test code with the `pge/c/interpen.c` source code
  - does it work?

interpen.h

```
/* automatically created by mc from ../git-pge/m2/interpen.def.  */


#if !defined (_interpen_H)
#   define _interpen_H

#   ifdef __cplusplus
extern "C" {
#   endif
#   if !defined (PROC_D)
#      define PROC_D
       typedef void (*PROC_t) (void);
       typedef struct { PROC_t proc; } PROC;
#   endif
```


interpen.h

```
#   include "Gcoord.h"
#   include "Gsegment.h"
#   include "Ghistory.h"

#   if defined (_interpen_C)
#      define EXTERN
#   else
#      define EXTERN extern
#   endif

typedef struct interpen_interCircle_r interpen_interCircle;

struct interpen_interCircle_r {
                               double radius;
                               coord_Coord center;
                             };
```


interpen.h

```
/*
   segmentCollide - returns TRUE if segment, a, overlaps with, b.
                    If true is returned then collisionPoint will
                    be set to the intersection point.
*/

EXTERN unsigned int interpen_segmentsCollide (segment_Segment a,
                                              segment_Segment b,
                                              coord_Coord *p,
                                              history_whereHit *at0
                                              history_whereHit *at1
                                              unsigned int *ptn0,
                                              unsigned int *ptn1);
```

interpen.h

```
/*
   circleCollide - return TRUE if circles, a, b, collide.
*/

EXTERN unsigned int interpen_circleCollide (interpen_interCircle a,
                                            interpen_interCircle b)
```


interpen.h

```
/*
   circleSegmentCollide - Pre-condition:  interCirle, c, and Segmen
                          s, are well formed.
                          Post-condition:  return TRUE if circle, c
                          collides with segment, s.
                          If true is returned then the, point,
                          on the line in deepest collision
                          with the circle is filled in and likewise
                          at, is set to corner or edge.
                          Indicating which part of the segment
                          collides with the circle.
                          ptn will be set to 0 if point1 of the
                          segment collides with the circle.
                          ptn will be set to 1 if point2 of the
                          segment collides with the circle.
*/

EXTERN unsigned int interpen_circleSegmentCollide (interpen_interCi
                                                   segment_Segment
                                                   coord_Coord *poi
                                                   history_whereHit
                                                   unsigned int *pt
```


interpen.h

```
/*
   initCircle - create and return an interpen_interCircle structure
*/

EXTERN interpen_interCircle interpen_initCircle (double radius,
                                                 coord_Coord point)

/*
   unitTest - run some basic unit tests for the module.
*/

EXTERN void interpen_unitTest (void);
#   ifdef __cplusplus
}
#   endif

#   undef EXTERN
#endif
```


circleCollide algorithm

- sum the radii of both circles
- subtract the centers one circle from the other
  - thus generating a vector, distance, from one circle to the second circle
- `return vector_length (distance) <= sum`

circleCollide

- **$HOME/Sandpit/git-pge/c/interpen.c**

```
unsigned int interpen_circleCollide (interpen_interCircle
                                     interpen_interCircle
{
  double radiusSum;
  coord_Coord distance;

  radiusSum = a.radius+b.radius;
  distance = coord_subCoord (a.center, b.center);
  return (coord_lengthCoord (distance)) <= radiusSum;
}
```


circlePointCollide algorithm

- compute the distance vector between the point and circle centre
- `return vector_length (distance) <= radius of circle`

circlePointCollide

- **$HOME/Sandpit/git-pge/c/interpen.c**

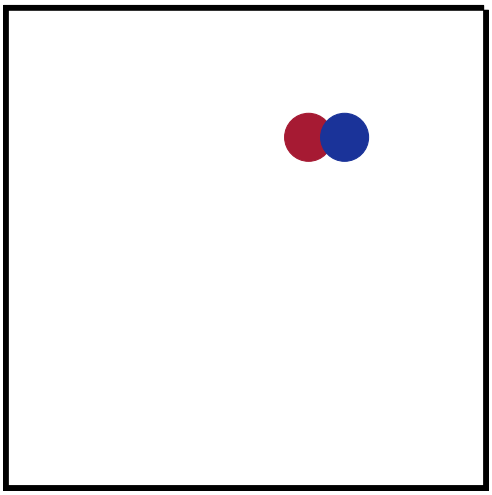
```
/*
   circlePointCollide - returns TRUE if circle, c,
                        and point, p, collide.
*/

static unsigned int circlePointCollide (interpen_interCir
                                        coord_Coord p)
{
  coord_Coord distance;

  distance = coord_subCoord (c.center, p);
  return (coord_lengthCoord (distance)) <= c.radius;
}
```


Circle circle collisions

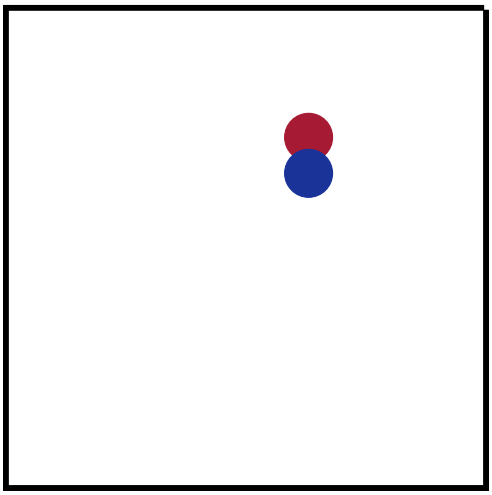
-



Circle circle collisions

- the two circles here are at:
  - `(6, 7)  radius .5  darkred`
  - `(6.75, 7)  radius .5  darkblue`
- we could use this test as a regression test (unit test) for `interpen.c`

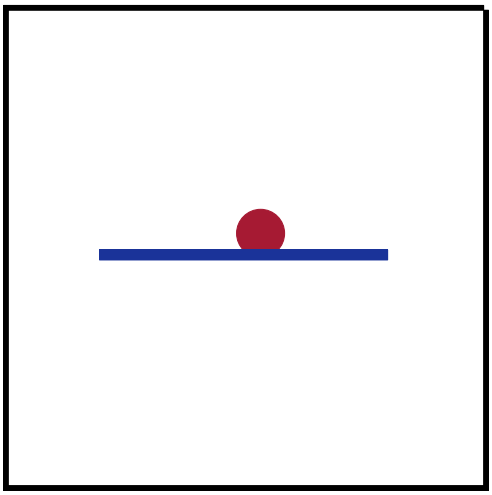
Circle circle collisions



Circle circle collisions

- the two circles here are at:
  - `(6, 7)  radius .5  darkred`
  - `(6, 6.25)  radius .5  darkblue`
- we could use this test as a regression test (unit test) for `interpen.c`

circle segment collision test code diagram



circle segment collision test code diagram

- the circle is at position `.5, .5` radius `.05`
- the line is from `(.2,.48)` to `(.6,.48)`
- lookup the definition for `circleSegmentCollide` and write some test code for this function

Ginterpen.h snippet

- **pge/c/Ginterpen.h**

```
/*
   circleSegmentCollide -

   Pre-condition:  interCirle, c, and Segment, s, are well formed.
   Post-condition:  return TRUE if circle, c, collides with segment
   If true is returned then the, point, on the line in deepest coll
   with the circle is filled in and likewise, at, is set to corner
   Indicating which part of the segment collides with the circle.
   ptn will be set to 0 if point1 of the segment collides with the
   ptn will be set to 1 if point2 of the segment collides with the
*/

EXTERN unsigned int interpen_circleSegmentCollide
   (interpen_interCircle c, segment_Segment s, coord_Coord *point,
    history_whereHit *at, unsigned int *ptn);
```


Ginterpen.h snippet

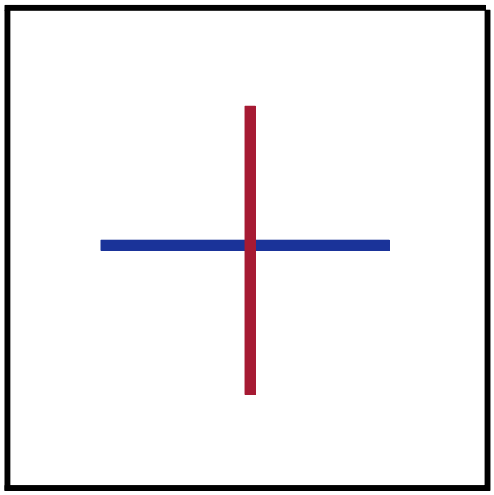
- the function takes 5 parameters:
  - the first two are the circle and segment in question
  - `point` is the deepest point of the line in the circle (if they are in collision)
  - `at` describes the `point` (whether it was the `corner`/end of the segment or `edge`/midpoint)
  - `ptn` is the point number of the segment which is in collision (assuming `at==corner`) and will be either `0` or `1` representing the first or second point in the segment
- write some test code for this function - and check all possible parameters

segmentsCollide

```
/*
  segmentCollide - returns TRUE if segment, a, overlaps w
     If true is returned then, p, will be set to
     the intersection point.  ata, atb determine where se
     and segment, b, hit (corner or edge).  If ata == cor
     ptna is either 0 or 1 representing the point which c
     Likewise if atb == corner then
     ptna is either 0 or 1 representing the point which c
*/

EXTERN unsigned int interpen_segmentsCollide
     (segment_Segment a, segment_Segment b, coord_Coord *
      history_whereHit *ata, history_whereHit *atb,
      unsigned int *ptna, unsigned int *ptnb);
```

segmentsCollide



segmentsCollide

- in the previous example the two segments are defined by the end points
  - blue `(2,5)` and `(8,5)`
  - red `(5,2)` and `(5,8)`
- write some test code to check the function behaves as expected
  - write two tests, one with the values as above and another with the end of a segment colliding with the other segment
- we will be using these functions in the construction of the coursework - to implement free moving polygons in pge