

# How to start implementing frozen bubble in PGE

- start by examining the bagatelle example  
`examples/bagatelle/bagatelle.py`
- remove triangles, silos and pegs

## How to solve the bubble freezing and unfreezing?

- these are code hints, they will need polishing and completing
  - this code will contain bugs and has not been tested
  - but it maybe of some help!

```
#  
# the bubble class contains the pge object (circle) and colour  
# and a list of children (bubbles which have frozen onto us)  
#  
  
class bubble:  
    def __init__ (self, cir, col):  
        self.circle = cir  
        self.colour = col  
        self.children = [] # which bubbles are connected to us?  
        cir.fix () # we need to fix this moving circle  
        # a circle with mass cannot be fixed in pge --fixme--
```

## How to solve the bubble freezing and unfreezing?

```
■ bubbleDict = {} # allows us to obtain a bubble list from a pge circle
currentCircle = None
currentColour = None
rootBubbles = [] # bubbles attached to the top bar
frozenBubbles = [] # all frozen bubbles
```

## How to solve the bubble freezing and unfreezing?

```
#  
# bubble_hits_bar - the callback used when the bubble hits  
#                   the top of frame. We create a bubble  
#                   from the circle and colour and place it  
#                   into the dictionary and as a root bubble.  
#  
  
def bubble_hits_bar (o, e):  
    global rootBubbles, frozenBubbles, bubbleDict  
    b = bubble (currentCircle, currentColour)  
    bubbleDict[currentCircle] = [b]  
    rootBubbles += [b]  
    frozenBubbles += [b]
```

## How to solve the bubble freezing and unfreezing?

```
def bubble_hits_bubble (o, e):  
    p = e.collision_between ()  
    if p != None and p != []:  
        for c in p:  
            if c != currentCircle:  
                addBubble (c)
```

## How to solve the bubble freezing and unfreezing?

```
def addBubble (c):  
    blist = bubbleDict[c]  
    if blist[0].colour == currentColour:  
        if len (blist) > 3:  
            unfreeze (blist)  
        else:  
            addCurrent (c)  
    else:  
        addCurrent (c)
```

## How to solve the bubble freezing and unfreezing?

```
def addCurrent (c):  
    b = bubble (currentCircle, currentColour)  
    b.children += [bubbleDict[c]]  
    bubbleDict[currentCircle] = [b]  
    frozenBubbles += [b]
```

## unfreeze (version 1)

```
def unfreeze (blist):  
    for b in blist:  
        print "circle", b, "should be unfixed"  
        b.unfix () # --fixme-- this is unimplemented in pge
```

- does not solve if any bubbles are frozen underneath the `blist`



## unfreeze (version 2)

```
def unfreeze (blist):  
    global rootBubbles, frozenBubbles  
    moving = []  
    for b in frozenBubbles:  
        moving += [b]  
    # place all into the moving list  
    for b in blist:  
        if b in rootBubbles:  
            rootBubbles.remove (b)  
        if b in frozenBubbles:  
            frozenBubbles.remove (b)
```

## unfreeze (version 2)

```
for b in rootBubbles:  
    moving = refreeze (moving, b)  
# now unfix all moving bubbles  
for b in moving:  
    b.circle.unfix ()  
    b.children = []
```

## refreeze

```
def refreeze (moving, b):  
    if b in moving:  
        moving.remove (b)  
        for c in b.children:  
            moving = refreeze (moving, c)  
    return moving
```

## unfix

- it would be useful to implement `unfix` in `pge`
  
- in order to achieve this we need to modify the following files
  - `pge/i/pgeif.i`
  - `pge/c/Gpgeif.h`
  - `pge/c/pgeif.c`
  - `pge/c/GtwoDsim.h`
  - `pge/c/twoDsim.c`

## Layers and source files to be altered

- `pge/python/pge.py`
  - the user level python API file
  - this is the only PGE visible file to the user
  
- `pge/i/pgeif.i`
  - the swig interface (python calling C/C++ definition)
  - remember to edit both sections (C/C++ section and the Python section)
  - hint look for `%{` and `}%` delimiters

## Layers and source files to be altered

- `pge/c/Gpgeif.h`
  - header file for `pgeif.c`
  - contains the external functions implemented inside `pgeif.c`
  
- `pge/c/pgeif.c`
  - its purpose is to allow, colours, polygons, circles, springs, to be given a unique integer
  - thereafter all references to objects will be achieved via the objects, `id`.
  - notice that inside `pge/c/twoDsim.c` colours and circles are different
  
- `pge/c/GtwoDsim.h` contains the prototypes and external declarations for `pge/c/twoDsim.c`

## unfix hints

- in the five files mentioned on the previous slides
  - search for the `fix` function
  - duplicate it, and change the duplicate to `unfix`
  - adjust the implementation of the function (if necessary)
  - finally correct the comments!
  
- note that the file `pge/python/pge.py` will need a few edits to change the runtime checking
  - it currently checks to ensure that a `fixed` object has no mass
  - and a `fixed` object can not be given an acceleration/velocity
  - obviously this needs to be changed
  - in effect these checks need to be removed

## pge/python/pge.py

- removing the checking code
  - the checking code is found in many methods and will look similar to

```
def velocity (self, vx, vy):  
    ...  
    self._check_not_fixed ("assign a velocity")  
    ...
```



## Running your pge code

■ `$ ./localrun.sh ../pge/frozenbubble/frozenbubble.py`