Making a bomb

We are now going to place a single bomb on the screen. This will be a ball. It will explode and when other balls hit it, they also will explode.

Exploding and shrinking

First add the following functions to your `ball.py` file:

```python
def explode(self):
    if self.radius < self.BALL_EXPLODE_RADIUS:
        self.radius += self.BALL_EXPLODE_SPEED

    # to hold for a time before shrinking it
    self.time_exploded += 1
    if self.time_exploded > self.BALL_TIME_EXPLODING:
        self.exploding = False
        self.shrinking = True

def shrink(self):
    if self.radius > 0:
        self.radius -= self.BALL_EXPLODE_SPEED

    if self.radius <= 0:
        self.set_inactive()  # will later remove from list
```

One of the functions is going to take care of making the balls explode, the other will take care of making them shrink. Note that they rely on several variables, `BALL_EXPLODE_SPEED`, `BALL_TIME_EXPLODING` and `BALL_EXPLODE_RADIUS`. You should add lines that will set all those inside of your `__init__` function at the top of `ball.py`. (Some lines might already be there)

I added these two lines:

```python
self.BALL_EXPLODE_RADIUS = 3 * self.radius          # we will let the ball explode to 3
times its current size
self.BALL_EXPLODE_SPEED = 1
```

Immediately after the line that sets the radius in the `__init__` function.

Now, let's test it. In your `__init__` function (in `ball.py`), search for the line:

```python
self.exploding = False
```

Change the `False` to a `True`. This means that for every ball we create, we want to immediately make it explode. Run your `main.py` again. You should be able to place balls that will immediately grow and after a time, they will shrink to nothingness. (You might want to change the `self.BALL_TIME_EXPLODING` variable to see how that affects the ball).

Note that your ball will almost shrink to nothingness, in fact it is still a pixel. We do not want the balls to exist after they shrink to nothingness. Find your `evolve` method inside of `BallData` and find the loop in that function that looks like this:

```python
# move all the balls
for ball in self.balls:
    ball.evolve(self.width, self.height, self.cell_size)
```

Change it to this:

```python
# move all the balls
for ball in self.balls:
    if not ball.is_active():  # this will test to see if
        self.balls.remove(ball)  # and remove it from the
game
    ball.evolve(self.width, self.height, self.cell_size)
```
Test your main again and this time, the ball should REALLY shrink to nothingness. Remember, we were just testing though, so now change your `self.exploding` variable back to `False` by default.

**Making the explosion occur when the ball collides**

If you followed an earlier tutorial, when the balls hit each other, they probably will **bounce**. We are going to change this so that instead of bouncing, the balls blow up!

Navigate to the `evolve` method of `[BallData]` again. Look for a loop that looks like this:

```python
# check to see if any ball collides with any other
for ball in self.balls:
    for otherBall in self.balls:
        if ball == otherBall:  # we don't want to compare ball with itself
            continue
        else:
            if ball.intersects(otherBall):
                ball.bounce(otherBall)  # what to do when they DO hit
```

If you don’t have the above code, put it in now. We need to change the very last line of code (that calls the bounce function), to this:

```python
if ball.intersects(otherBall):
    if not ball.is_exploding() and not ball.is_shrinking():                     # new things
to do
    ball.exploding = True                                                   # new
    if not otherBall.is_exploding() and not otherBall.is_shrinking():           # new
        otherBall.exploding = True                                              # new
```

Add the following methods anywhere within the `ball.py` file.

```python
def is_exploding(self):
    return self.exploding
```

Now, as you run the program you should see that balls, grow and shrink when they hit another ball. They should stop moving when they do this.

**Placing a bomb**

We don’t want to trigger the chain reaction above until and unless the user has placed a bomb on the screen. That is what we will try to do now. We will begin by adding another method to the `ball.py` file. Anywhere is fine:

```python
def set_position(self, position):
    self.position = position
```

This method allows us to manually fix the position of any ball... We will only use it for our bomb though.

Now, make sure your that the first few lines of your `evolve` method in `[BallData]`, look like this:

```python
def evolve(self, keys, newkeys, buttons, newbuttons, mouse_position):
    if not self.evolving:
        return
```

We also need to tell our `[Bomb]` to be drawn to the screen. Before the `return` in `[BallData]` `draw` function. Add
the following line:

```python
self.bomb.draw(surface, self.cell_size)
```

Next, create a new instance of a `Bomb` by putting this line anywhere inside the `newGame` method (of `BallData`):

```python
self.bomb = Ball((0,0))
# this will create a new ball at position 0,0
```

Finally, in `BallData.py`, find the line in the `__init__` method where the mouse is set to invisible. Remove the `#` at the front of the line to uncomment it and make it active.

```python
pygame.mouse.set_visible(False)
```

Hopefully if you run it at this point, you will now have a ball for your cursor and you will be able to create balls when you click. When the balls collide, they will grow and then shrink.

### Finishing touches

I am going to make my game start out with a fixed number of balls. When the mouse button is clicked, we will not spawn a new ball, but we will freeze our bomb and make it explode. The idea is to have all the balls exploded with a single bomb.

Find the loop that is commented out inside the `newGame` function within `BallData`. Uncomment it and make sure it is indented appropriately. This will give me a fixed number of balls, in random locations throughout the screen. (You can tweak the `self.num_balls` variable to adjust the number of balls). We can also make them a more uniform size (which makes our game look better), by changing the BALL_MIN_RADIUS to 1 and the BALL_MAX_RADIUS to 2 in `ball.py`.

To remove spawning balls when the mouse is clicked, find the following code in the `evolve` method, and comment out the last 2 lines. Instead, we will add a line that will freeze the bomb in place and cause it to start exploding.

```python
if 1 in newbuttons:
    b = Ball((x,y))
# comment this line out
    self.balls.append(b)
# and this one
    self.bomb.exploding = True
# add this line
```

We need to make sure to always `evolve` our bomb. Add this line to the very bottom of the `BallData evolve` method:

```python
self.bomb.evolve(self.width, self.height, self.cell_size)
```

Test your code again by running it. Whoops! The balls are still growing and shrinking immediately. We don’t want them to do this unless they have collided with a ball that is actually exploding. Let’s fix that now.

Find the following `if` statement in the `evolve` method:

```python
if ball.intersects(otherBall):
```

Change it to the following (this will make sure that we only explode if we have hit something else that is currently shrinking or exploding):

```python
if ball.intersects(otherBall) and (ball.is_exploding() or ball.is_shrinking() or otherBall.is_exploding() or otherBall.is_shrinking()):
```

Also, a few lines above that, you should see a for loop that starts like this:

```python
for ball in self.balls:
```

You should add the following lines immediately after:

```python
if ball.intersects(self.bomb) and (self.bomb.is_exploding() or self.bomb.is_shrinking()):
```
This will test to see if the ball has hit the bomb. Whew. Run it again. Hopefully, you have a working game.

**Extra fun**

- Multiple bombs
- Different kinds of bombs
- The bomb might have shrunk to a single pixel. You can tell `ball.py` not to paint things after they are complete.