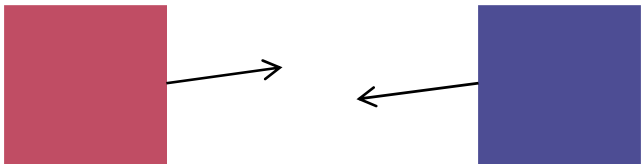
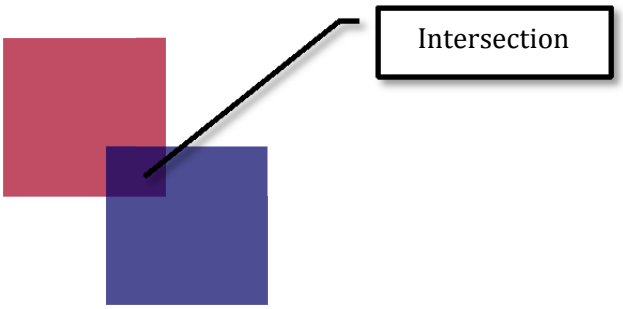
PART FOUR – COLLISION DETECTION

The final part of this set of tutorials concerns collision detection.

Consider two sprites, A and B moving in the directions indicated by the arrows.



We can know a collision has occurred when the polygons representing the two intersect as demonstrated below.



We could calculate this manually by comparing the x and y co-ordinates of each vertice on each polygon. Fortunately, a readily available method has been provided in JavaFX.

In order to test if I have been able to hit the baddie I will have to loop through the ArrayList, get a polygon for each missile and check it for intersections with the polygon that represents the baddie. If I have displayed that level of skill then I will clear the screen and display a victory message.

This is my final checkHit() method. I have called this *before* I update my missile bank (in the ActionListemer). It is also only worth calling both this and updateMissiles() if there are missiles currently being fired (i.e. if missileBank.size()>0).

```
void checkHit(){
   for (int i = 0; i<missileBank.size(); i++){
      Missile m  = (Missile)missileBank.get(i);
      if (m.getImageView().getBoundsInParent().intersects(
                  baddie.getImageView().getBoundsInParent())){
         missileBank.clear();
         g.getChildren().clear();
         Image won = new Image(getClass().getResourceAsStream("victory.png"));
         g.getChildren().add(new ImageView(won));
      }
   }
```

```
}
```

```
if (m.getImageView().getBoundsInParent().intersects(
                baddie.getImageView().getBoundsInParent())){
```

The condition for this if-statement gets the Rectangle that represents the space taken up by the current missile sprite and checks to see if this intersects with the Rectangle that represents the space taken up by the baddie.

```
missileBank.clear();
g.getChildren().clear();
```

This clears the ArrayList of missiles and the root group so the stage is clear and is followed by the creation and addition of a victory background.

If you run your game, you should be able to 'win' the game by hitting the baddie with a missile.

There are a number of improvements that can be made to this game. Some of them start off with hiding members of each class that are not needed to be visible by using the keyword private (for example private int x when declaring the global variables). For further improvements and extensions see the final file in this tutorial series.