Part 05

# Pygame Zero Invaders II

Space Invaders must be the first computer game that springs to mind for a lot of people. Here in part two we will take our basic game from part one and add all the extras

MAKER

**Mark Vanstone**

Educational software author from the nineties, author of the ArcVenture series, disappeared into the corporate software wasteland. Rescued by the Raspberry Pi!

magpi.cc/YiZnxl

@mindexplorers

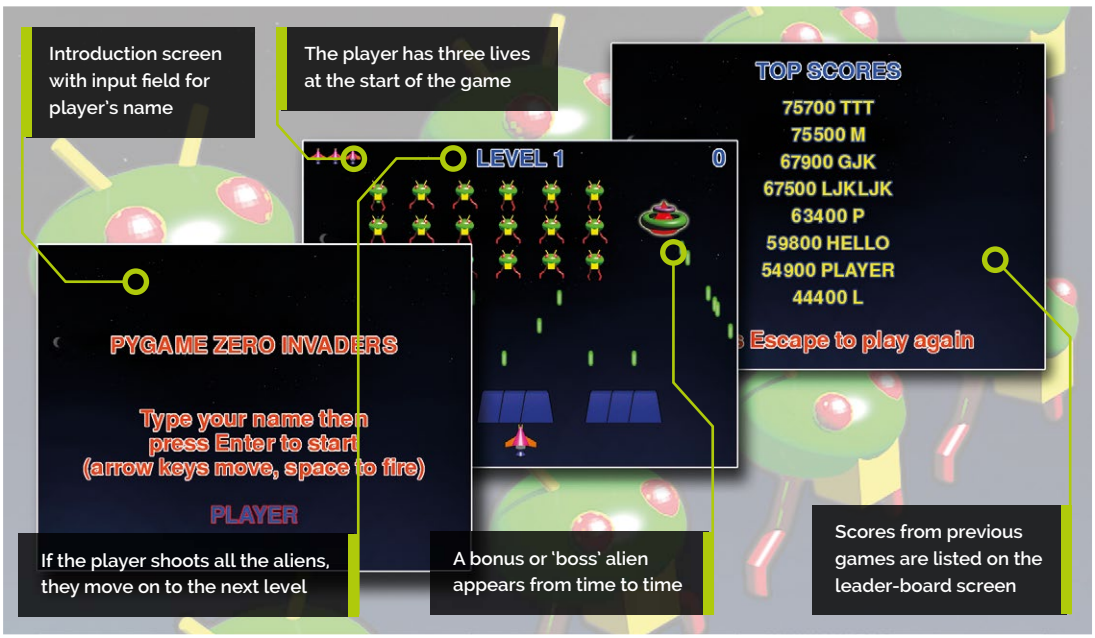
In part one, last issue, we set up the basics for our Space Invaders game with our player ship controlled by the keyboard, defence bases, the aliens moving backwards and forwards across the screen, and lasers flying everywhere. In this part we will add lives and levels to the game, introduce a bonus alien, code a leader board for high scores, and add some groovy sound effects. We may even get round to adding an introduction screen if we get time. We are going to start from where we left off in part one. If you don't have the part one code and files, you can download them from GitHub at **magpi.cc/JxQrdd**.

## 01 You only live thrice

It was a tradition with Space Invaders to be given three lives at the start of the game. We can easily set up a place to keep track of our player lives by writing `player.lives = 3` in our `init()` function. While we are in the `init()` function, let's add a player name variable with `player.name = ""` so that we can show names on our leader board, but we'll come to that in a bit. To display the number of lives our player has, we can add `drawLives()` to our `draw()` function and then define our `drawLives()` function containing a loop which 'blits' **life.png** once for each life in the top left of the screen.

## You'll Need

- An image manipulation program such as GIMP, or images from **magpi.cc/mYSswY**
- The latest version of Pygame Zero (1.2)
- The Audacity sound editor or similar or sounds available from **magpi.cc/mYSswY**
- Speakers or headphones



## 02 Life after death

Now we have a counter for how many lives the player has, we will need to write some code to deal with what happens when a life is lost. In part one we ended the game when the `player.status` reached 30. In our `update()` function we already have a condition to check the `player.status` and if there are any aliens still alive. Where we have written `if player.status == 30:` we can write `player.lives -=1`. We can also check to see if the player has run out of lives when we check to see if the **RETURN** (aka **ENTER**) key is pressed.

## 03 Keep calm and carry on

Once we have reduced `player.lives` by one and the player has pressed the **RETURN** key, all we need to do to set things back in motion is to set `player.status = 0`. We may want to reset the laser list too, because if the player was hit by a flurry of lasers we may find that several lives are lost without giving the player a chance to get out of the way of subsequent lasers. We can do this by writing `lasers = []`. If the player has run out of lives at this point, we will send them off to the leader-board page. See **figure1.py** to examine the code for dealing with lives.

## 04 On the level

The idea of having levels is to start the game in an easy mode; then, when the player has shot all the aliens, we make a new level which is a bit harder than the last. In this case we are going to tweak a few variables to make each level more difficult. To start, we can set up a global variable `level = 1` in our `init()` function. Now we can use our `level` variable to alter things as we increase the value. Let's start by speeding up how quickly the aliens move down the screen as the level goes up. When we calculate the `movey` value in `updateAliens()`, we can write `movey = 40 + (5*level)` on the condition that `moveSequence` is 10 or 30.

## 05 On the up

To go from one level to the next, the player will need to shoot all the aliens. We can tell if there are any aliens left if `len(aliens) = 0`. So, with that in mind, we can put a condition in our `draw()` function with if `len(aliens) == 0:` and then draw text on the screen to say that the level has been cleared. We can put the same condition in the section of the `update()` function where we are waiting for **RETURN** to be pressed. When **RETURN** is pressed and the length of the aliens list is 0, we can add 1 to `level` and call `initAliens()` and `initBases()` to set things ready to start the new level.

## figure1.py

```
001. def draw()
002.     # additional drawing code
003.     drawLives()
004.     if player.status >= 30:
005.         if player.lives > 0:
006.             drawCentreText(
     "YOU WERE HIT!\nPress Enter to re-spawn")
007.         else:
008.             drawCentreText(
     "GAME OVER!\nPress Enter to continue")
009.
010. def init():
011.     # additional player variables
012.     player.lives = 3
013.     player.name = ""
014.
015. def drawLives():
016.     for l in range(player.lives):
017.         screen.blit("life", (10+(l*32),10))
018.
019. def update():
020.     # additional code for life handling
021.     global player, lasers
022.     if player.status < 30 and len(aliens) > 0:
023.         if player.status > 0:
024.             player.status += 1
025.             if player.status == 30:
026.                 player.lives -= 1
027.     else:
028.         if keyboard.RETURN:
029.             if player.lives > 0:
030.                 player.status = 0
031.                 lasers = []
032.             else:
033.                 # go to the leader-board
034.                 pass;
035.
036. def drawCentreText(t):
037.     screen.draw.text(t , center=(400, 300), owidth=0.5,
     ocolor=(255,255,255), color=(255,64,0) , fontsize=60)
```

▲ Code to deal with player lives. Notice the `drawCentreText()` function to short-cut printing text to the centre of the screen

## figure2.py

```
001.  def updateBoss():
002.      global boss, level, player, lasers
003.      if boss.active:
004.          boss.y += (0.3*level)
005.          if boss.direction == 0: boss.x -= (1* level)
006.          else: boss.x += (1* level)
007.          if boss.x < 100: boss.direction = 1
008.          if boss.x > 700: boss.direction = 0
009.          if boss.y > 500:
010.              sounds.explosion.play()
011.              player.status = 1
012.              boss.active = False
013.          if randint(0, 30) == 0:
014.              lasers.append(Actor("laser1",
      (boss.x,boss.y)))
015.              lasers[len(lasers)-1].status = 0
016.              lasers[len(lasers)-1].type = 0
017.      else:
018.          if randint(0, 800) == 0:
019.              boss.active = True
020.              boss.x = 800
021.              boss.y = 100
022.              boss.direction = 0
```

▲ Code to update the boss or bonus alien. This code runs when the boss is active or uses random numbers to see if it's time to make it active

### 06 Front and centre

You may have noticed in **figure1.py** that we made a couple of calls to a function called `drawCentreText()` which we have not yet discussed. All that this function does is to shorten the process of writing text to the centre of the screen. We assume that the text will be positioned at coordinates (400, 300) and will have a set of standard style settings and colours, and the function definition just contains one line: `screen.draw.text(t , center=(400, 300), owidth=0.5, ocolor=(255,255,255), color=(255,64,0), fontsize=60)` – where `t` is passed into the function as a parameter.

### 07 Flying like a boss

To liven up our game a little bit, we are going to add in a bonus or boss alien. This could be triggered in various ways, but in this case we will start the boss activity with a random number. First we will need to create the boss actor. Because there will only ever be one boss alien on screen at any time, we can just use one actor created near the start of our code. In this case we don't need to give it coordinates as we will start the game with the boss actor not being drawn. We write `boss = Actor("boss")`.


▲ Lasers can be very bad for your health. Best to avoid them

### 08 Keeping the boss in the loop

We want to start the game with the boss not being displayed, so we can add to our `init()` function `boss.active = False` and then in our `draw()` function `if boss.active: boss.draw()`, which will mean the boss will not be drawn until we make it active. In our `update()` function, along with our other functions to update elements, we can call `updateBoss()`. This function will update the coordinates of the boss actor if it is active or, if it is not, check to see if we need to start a new boss flying. See **figure2.py** for the `updateBoss()` function.

### 09 Did you hear that?

You may have noticed that in **figure2.py** we have an element of Pygame Zero that we have not discussed yet, and that is sound. If we write `sounds.explosion.play()`, then the sound file located at **sounds/explosion.wav** will be played. There are many free sound effects for games on the internet. If you use a downloaded WAV file, make sure that it is fairly small. You can edit WAV sound files with programs like Audacity. We can add sound code to other events in the program in the same way, like when a laser is fired.

### 10 More about the boss

Staying with **figure2.py**, note how we can use random numbers to decide when the boss becomes active and also when the boss fires a laser. You can change the parameters of the `randint()` function to alter the occurrence of these

events. You can also see that we have a simple path calculating system for the boss to make it move diagonally down the screen. We use the `level` variable to alter aspects of the movement. We treat the boss lasers in the same way as the normal alien lasers, but we need to have a check to see if the boss is hit by a player laser. We do this by adding a check to our `checkPlayerLaserHit()` function.

## 11 Three strikes and you're out

In the previous episode, the game ended if you were hit by a laser. In this version we have three chances before the game ends, and when it does, we want to display a high score table or leader board to be updated from one player to the next. There are a few considerations to think about here. We need a separate screen for our leader board; we need to get players to enter their name to put against each score and we will have to save the score information. In other programs in this series we have used the variable `gameStatus` to control different screens, so let's bring that back for this program.

## 12 Screen switching with gameStatus

We will need three states for the `gameStatus` variable. If it is set to 0 then we should display an intro screen where we can get the player to type in their name. If it is set to 1 then we want to run code for playing the game. And if it is set to 2 then we display the leader-board page. Let's first deal with the intro screen. Having set our variable to 0 at the top of the code, we need to add a condition to our `draw()` function: `if gameStatus == 0:`. Then, under that, use `drawCentreText()` to show some intro text and display the `player.name` string. To start with, `player.name` will be blank.

## 13 A name is just a name

Now to respond to the player typing their name into the intro screen. We will write a very simple input routine and put it in the built-in Pygame Zero function `on_key_down()`. **figure3.py** shows how we do this. With this code, if the player presses a key, the name of the key is added to the `player.name` string unless the key is the **BACKSPACE** key, in which case we remove the last character. Notice the rather cunning way of doing that with `player.name = player.name[:-1]`. We also ignore the **RETURN** key, as we can deal with that in our `update()` function.

### figure3.py

```
001. def on_key_down(key):
002.     global player
003.     if gameStatus == 0 and key.name != "RETURN":
004.         if len(key.name) == 1:
005.             player.name += key.name
006.         else:
007.             if key.name == "BACKSPACE":
008.                 player.name = player.name[:-1]
```

▲ Code for capturing keyboard input for the player to input their name on the introduction screen

## 14 Game on

When the player has entered their name on the intro screen, all we need to do is detect a press of the **RETURN** key in our `update()` function and we can switch to the game part. We can easily do this by just writing `if gameStatus == 0:` and then under that, `if keyboard.RETURN and player.name != "": gameStatus = 1`. We will also now need to put our main game update code under a condition, `if gameStatus == 1:`. We will also need to have the same condition in the `draw()` function. Once this is done, we have a system for switching from intro screen to game screen.

## 15 Leader of the pack

So now we come to our leader-board screen. It will be triggered when the player loses the third life. When that happens, we set gameStatus to 2 and put a condition in our `draw()` and `update()` functions to react to that. When we switch to our leader board, we need to display the high score list – so, we can write in our `draw()` function: `if gameStatus == 2: drawHighScore()`. Going back to **figure1.py**, you'll see that we left a section at the end commented out, ready for the leader board. We can now fill this in with some code.

## 16 If only I learned to read and write

We are going to save all our scores in a file so that we can get them back each time the

## figure4.py

```
001.  def readHighScore():
002.      global highScore, score, player
003.      highScore = []
004.      try:
005.          hsFile = open("highscores.txt", "r")
006.          for line in hsFile:
007.              highScore.append(line.rstrip())
008.      except:
009.          pass
010.      highScore.append(str(score)+ " " + player.name)
011.      highScore.sort(key=natural_key, reverse=True)
012.
013.  def natural_key(string_):
014.      return [int(s) if s.isdigit() else s for s in
      re.split(r'(\d+)', string_)]
015.
016.  def writeHighScore():
017.      global highScore
018.      hsFile = open("highscores.txt", "w")
019.      for line in highScore:
020.          hsFile.write(line + "\n")
021.
022.  def drawHighScore():
023.      global highScore
024.      y = 0
025.      screen.draw.text("TOP SCORES", midtop=(400, 30),
      owidth=0.5, ocolor=(255,255,255), color=(0,64,255) ,
      fontsize=60)
026.      for line in highScore:
027.          if y < 400:
028.              screen.draw.text(line, midtop=(400, 100+y),
      owidth=0.5, ocolor=(0,0,255), color=(255,255,0) ,
      fontsize=50)
029.              y += 50
030.      screen.draw.text("Press Escape to play again" ,
      center=(400, 550), owidth=0.5, ocolor=(255,255,255),
      color=(255,64,0) , fontsize=60)
```

▲ Code for reading, writing, sorting, and drawing the high score leader board

game is played. We can use a simple text file for this. When a new score is available, we will have to read the old score list in, add our new score to the list, sort the scores into the correct order, and then save the scores back out to create an updated file. So, the code we need to write in our `update()` function will be to call a `readHighScore()` function, set our `gameStatus` to 2, and call a `writeHighScore()` function.


▲ All the aliens have been destroyed. It's time to move up a level

### 17 Functions need to function

We have named three functions that need writing in the last couple of steps: `drawHighScore()`, `readHighScore()`, and `writeHighScore()`. Have a look at **figure4.py** to see the code that we need in these functions. The file reading and writing are standard Python functions. When reading, we create a list of entries and add each line to a list. We then sort the list into highest-score-first order. When we write the file, we just write each list item to the file. To draw the leader board, we just run through the high-score list that we have sorted and draw the lines of text to the screen.

### 18 Sort it out

It's worth mentioning the way we are sorting the high scores. In **figure4.py** we are adding a key sorting method to the list sorting function. We do this because the list is a string but we want to sort by the high score, which is numerical, so we break up the string and convert it to an integer and sort based on that value rather than the string. If we didn't do this and sorted as a string then all the scores starting with 9 would come first, then all the 8s, then all the 7s and so on, with 9000 being shown before 80 000, which would be wrong.

### 19 Well, that's all folks

That's about all we need for our Pygame Zero Invaders game other than all the additions that you could make to it. For example, you could have different graphics for each row of aliens. We're sure you can improve on the sounds that we have supplied, and there are many ways that the `level` variable can be worked into the code to make the different levels more difficult or more varied.

# invaderspart2.py

**DOWNLOAD THE FULL CODE:**
magpi.cc/gRWABf

```
import pgzrun, math, re, time
from random import randint
player = Actor("player", (400, 550))
boss = Actor("boss")
gameStatus = 0
highScore = []

def draw(): # Pygame Zero draw function
    screen.blit('background', (0, 0))
    if gameStatus == 0: # display the title page
        drawCentreText("PYGAME ZERO INVADERS\n\n\nType your name then\npress Enter to start\n(arrow keys move, space to fire)")
        screen.draw.text(player.name , center=(400, 500), owidth=0.5, ocolor=(255,0,0), color=(0,64,255) , fontsize=60)
    if gameStatus == 1: # playing the game
        player.image = player.images[math.floor(player.status/6)]
        player.draw()
        if boss.active: boss.draw()
        drawLasers()
        drawAliens()
        drawBases()
        screen.draw.text(str(score) , topright=(780, 10), owidth=0.5, ocolor=(255,255,255), color=(0,64,255) , fontsize=60)
        screen.draw.text("LEVEL " + str(level) , midtop=(400, 10), owidth=0.5, ocolor=(255,255,255), color=(0,64,255) , fontsize=60)
        drawLives()
        if player.status >= 30:
            if player.lives > 0:
                drawCentreText("YOU WERE HIT!\nPress Enter to re-spawn")
            else:
                drawCentreText("GAME OVER!\nPress Enter to continue")
        if len(aliens) == 0 :
            drawCentreText("LEVEL CLEARED!\nPress Enter to go to the next level")
    if gameStatus == 2: # game over show the leaderboard
        drawHighScore()

def drawCentreText(t):
    screen.draw.text(t , center=(400, 300), owidth=0.5, ocolor=(255,255,255), color=(255,64,0) , fontsize=60)

def update(): # Pygame Zero update function
    global moveCounter, player, gameStatus, lasers, level, boss
    if gameStatus == 0:
        if keyboard.RETURN and player.name != "":
            gameStatus = 1
    if gameStatus == 1:
        if player.status < 30 and len(aliens) > 0:
            checkKeys()
            updateLasers()
            updateBoss()
            if moveCounter == 0: updateAliens()
            moveCounter += 1
            if moveCounter == moveDelay:
                moveCounter = 0
            if player.status > 0:
                player.status += 1
                if player.status == 30:
                    player.lives -= 1
        else:
            if keyboard.RETURN:
                if player.lives > 0:
                    player.status = 0
                    lasers = []
                    if len(aliens) == 0:
                        level += 1
                        boss.active = False
                        initAliens()
                        initBases()
                else:
                    readHighScore()
                    gameStatus = 2
                    writeHighScore()
    if gameStatus == 2:
        if keyboard.ESCAPE:
            init()
            gameStatus = 0

def on_key_down(key):
    global player
    if gameStatus == 0 and key.name != "RETURN":
        if len(key.name) == 1:
            player.name += key.name
        else:
            if key.name == "BACKSPACE":
                player.name = player.name[:-1]

def readHighScore():
    global highScore, score, player
    highScore = []
    try:
        hsFile = open("highscores.txt", "r")
        for line in hsFile:
            highScore.append(line.rstrip())
    except:
        pass
    highScore.append(str(score)+ " " + player.name)
```

```
    highScore.sort(key=natural_key, reverse=True)

def natural_key(string_):
    return [int(s) if s.isdigit() else s for s in re.split(r'(\d+)', string_)]

def writeHighScore():
    global highScore
    hsFile = open("highscores.txt", "w")
    for line in highScore:
        hsFile.write(line + "\n")

def drawHighScore():
    global highScore
    y = 0
    screen.draw.text("TOP SCORES", midtop=(400, 30), owidth=0.5, ocolor=(255,255,255), color=(0,64,255) , fontsize=60)
    for line in highScore:
        if y < 400:
            screen.draw.text(line, midtop=(400, 100+y), owidth=0.5, ocolor=(0,0,255), color=(255,255,0) , fontsize=50)
            y += 50
    screen.draw.text("Press Escape to play again" , center=(400, 550), owidth=0.5, ocolor=(255,255,255), color=(255,64,0) , fontsize=60)

def drawLives():
    for l in range(player.lives):
        screen.blit("life", (10+(l*32),10))

def drawAliens():
    for a in range(len(aliens)): aliens[a].draw()

def drawBases():
    for b in range(len(bases)):
        bases[b].drawClipped()

def drawLasers():
    for l in range(len(lasers)): lasers[l].draw()

def checkKeys():
    global player, score
    if keyboard.left:
        if player.x > 40: player.x -= 5
    if keyboard.right:
        if player.x < 760: player.x += 5
    if keyboard.space:
        if player.laserActive == 1:
            sounds.gun.play()
            player.laserActive = 0
            clock.schedule(makeLaserActive, 1.0)
            lasers.append(Actor("laser2", (player.x,player.y-32)))
            lasers[len(lasers)-1].status = 0
            lasers[len(lasers)-1].type = 1
            score -= 100


def makeLaserActive():
    global player
    player.laserActive = 1

def checkBases():
    for b in range(len(bases)):
        if l < len(bases):
            if bases[b].height < 5:
                del bases[b]

def updateLasers():
    global lasers, aliens
    for l in range(len(lasers)):
        if lasers[l].type == 0:
            lasers[l].y += 2
            checkLaserHit(l)
            if lasers[l].y > 600:
                lasers[l].status = 1
        if lasers[l].type == 1:
            lasers[l].y -= 5
            checkPlayerLaserHit(l)
            if lasers[l].y < 10:
                lasers[l].status = 1
    lasers = listCleanup(lasers)
    aliens = listCleanup(aliens)

def listCleanup(l):
    newList = []
    for i in range(len(l)):
        if l[i].status == 0:
            newList.append(l[i])
    return newList

def checkLaserHit(l):
    global player
    if player.collidepoint((lasers[l].x, lasers[l].y)):
        sounds.explosion.play()
        player.status = 1
        lasers[l].status = 1
    for b in range(len(bases)):
        if bases[b].collideLaser(lasers[l]):
            bases[b].height -= 10
            lasers[l].status = 1

def checkPlayerLaserHit(l):
    global score, boss
    for b in range(len(bases)):
        if bases[b].collideLaser(lasers[l]):
            lasers[l].status = 1
    for a in range(len(aliens)):
        if aliens[a].collidepoint((lasers[l].x, lasers[l].y)):
            lasers[l].status = 1
            aliens[a].status = 1
            score += 1000
    if boss.active:
        if boss.collidepoint((lasers[l].x,
```

```
lasers[l].y)):
            lasers[l].status = 1
            boss.active = 0
            score += 5000

def updateAliens():
    global moveSequence, lasers, moveDelay
    movex = movey = 0
    if moveSequence < 10 or moveSequence > 30:
        movex = -15
    if moveSequence == 10 or moveSequence == 30:
        movey = 40 + (5*level)
        moveDelay -= 1
    if moveSequence >10 and moveSequence < 30:
        movex = 15
    for a in range(len(aliens)):
        animate(aliens[a], pos=(aliens[a].x
+ movex, aliens[a].y + movey), duration=0.5,
tween='linear')
        if randint(0, 1) == 0:
            aliens[a].image = "alien1"
        else:
            aliens[a].image = "alien1b"
            if randint(0, 5) == 0:
                lasers.append(Actor("laser1",
(aliens[a].x,aliens[a].y)))
                lasers[len(lasers)-1].status = 0
                lasers[len(lasers)-1].type = 0
                sounds.laser.play()
        if aliens[a].y > 500 and player.status ==
0:
            sounds.explosion.play()
            player.status = 1
            player.lives = 1
    moveSequence +=1
    if moveSequence == 40: moveSequence = 0

def updateBoss():
    global boss, level, player, lasers
    if boss.active:
        boss.y += (0.3*level)
        if boss.direction == 0:
            boss.x -= (1* level)
        else: boss.x += (1* level)
        if boss.x < 100: boss.direction = 1
        if boss.x > 700: boss.direction = 0
        if boss.y > 500:
            sounds.explosion.play()
            player.status = 1
            boss.active = False
        if randint(0, 30) == 0:
            lasers.append(Actor("laser1",
(boss.x,boss.y)))
            lasers[len(lasers)-1].status = 0
            lasers[len(lasers)-1].type = 0
    else:
        if randint(0, 800) == 0:
            boss.active = True
            boss.x = 800
            boss.y = 100
            boss.direction = 0

def init():
    global lasers, score, player, moveSequence,
moveCounter, moveDelay, level, boss
    initAliens()
    initBases()
    moveCounter = moveSequence = player.status =
score = player.laserCountdown = 0
    lasers = []
    moveDelay = 30
    boss.active = False
    player.images =
["player","explosion1","explosion2","explosion3",
"explosion4","explosion5"]
    player.laserActive = 1
    player.lives = 3
    player.name = ""
    level = 1

def initAliens():
    global aliens, moveCounter, moveSequence
    aliens = []
    moveCounter = moveSequence = 0
    for a in range(18):
        aliens.append(Actor("alien1", (210+
(a % 6)*80,100+(int(a/6)*64))))
        aliens[a].status = 0

def drawClipped(self):
    screen.surface.blit(self._surf, (self.x-32,
self.y-self.height+30),(0,0,64,self.height))

def collideLaser(self, other):
    return (
        self.x-20 < other.x+5 and
        self.y-self.height+30 < other.y and
        self.x+32 > other.x+5 and
        self.y-self.height+30 + self.height >
other.y
    )

def initBases():
    global bases
    bases = []
    bc = 0
    for b in range(3):
        for p in range(3):
            bases.append(Actor("base1",
midbottom=(150+(b*200)+(p*40),520)))
            bases[bc].drawClipped =
drawClipped.__get__(bases[bc])
            bases[bc].collideLaser =
collideLaser.__get__(bases[bc])
            bases[bc].height = 60
            bc +=1

init()
```