

# OBJECT ORIENTED PROGRAMMING WITH PYTHON

PROJECT BASED

2<sup>nd</sup> semester (Lect4)

*Project Idea*

*Project main development phases*

*First Phase*

*Class Diagram*


*Starting the Game Project*

*Creating VirusInvasion Class*

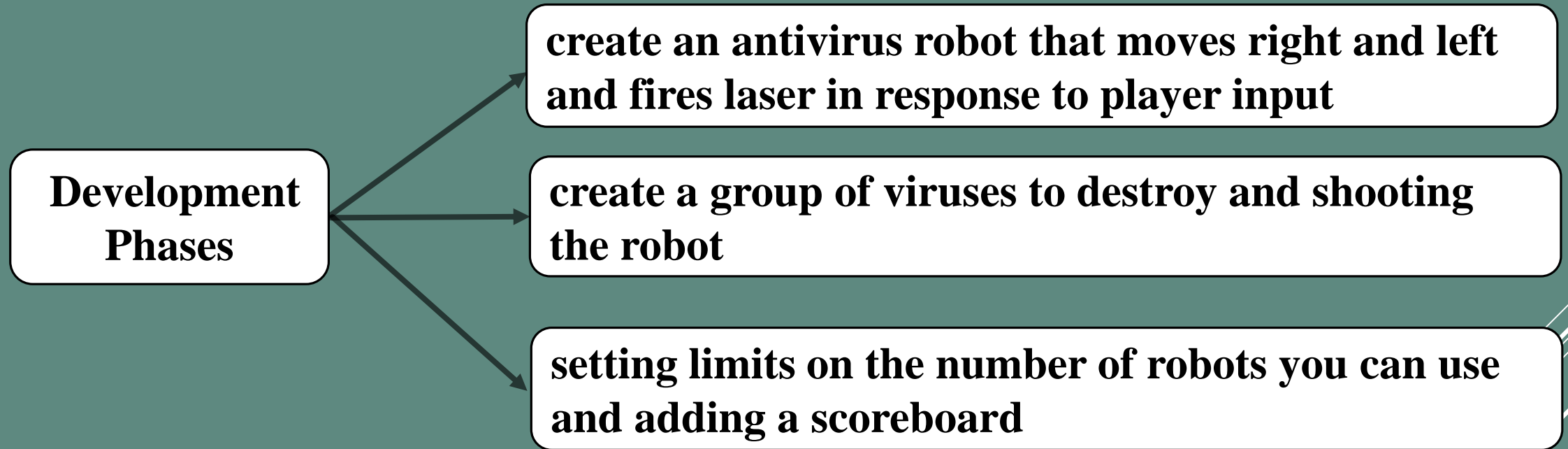
*Creating a Settings Class*

# *PROJECT IDEA*

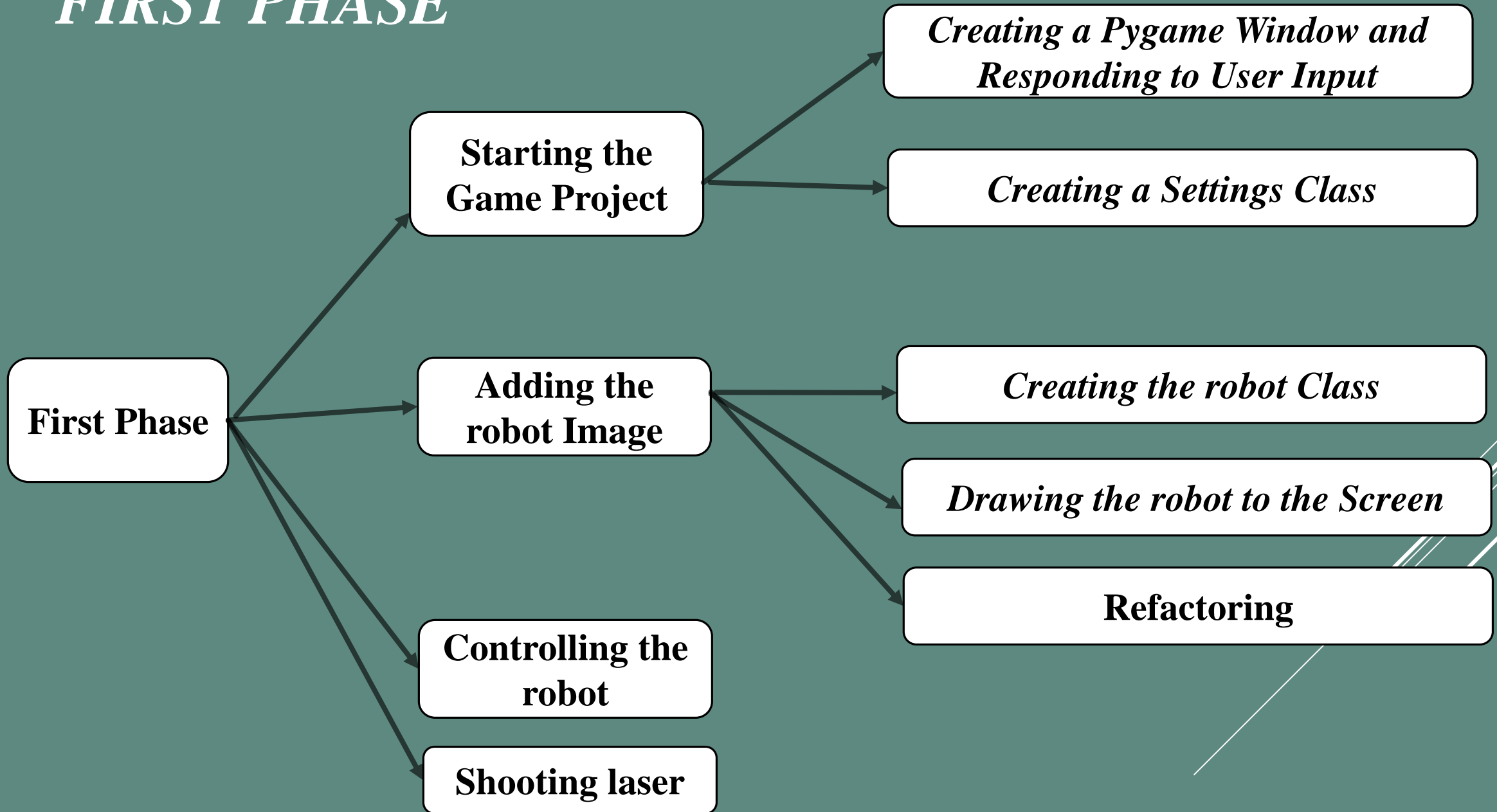
In this project we'll use the Pygame package to develop a 2D game in which the aim is to shoot up a group of viruses as they drop down the screen in levels that increase in speed and difficulty. At the end of the project, you'll have learned skills that will enable you to develop your own 2D games in Pygame.

A series of three parallel white lines of increasing length, slanted upwards from left to right, located in the bottom right corner of the slide.

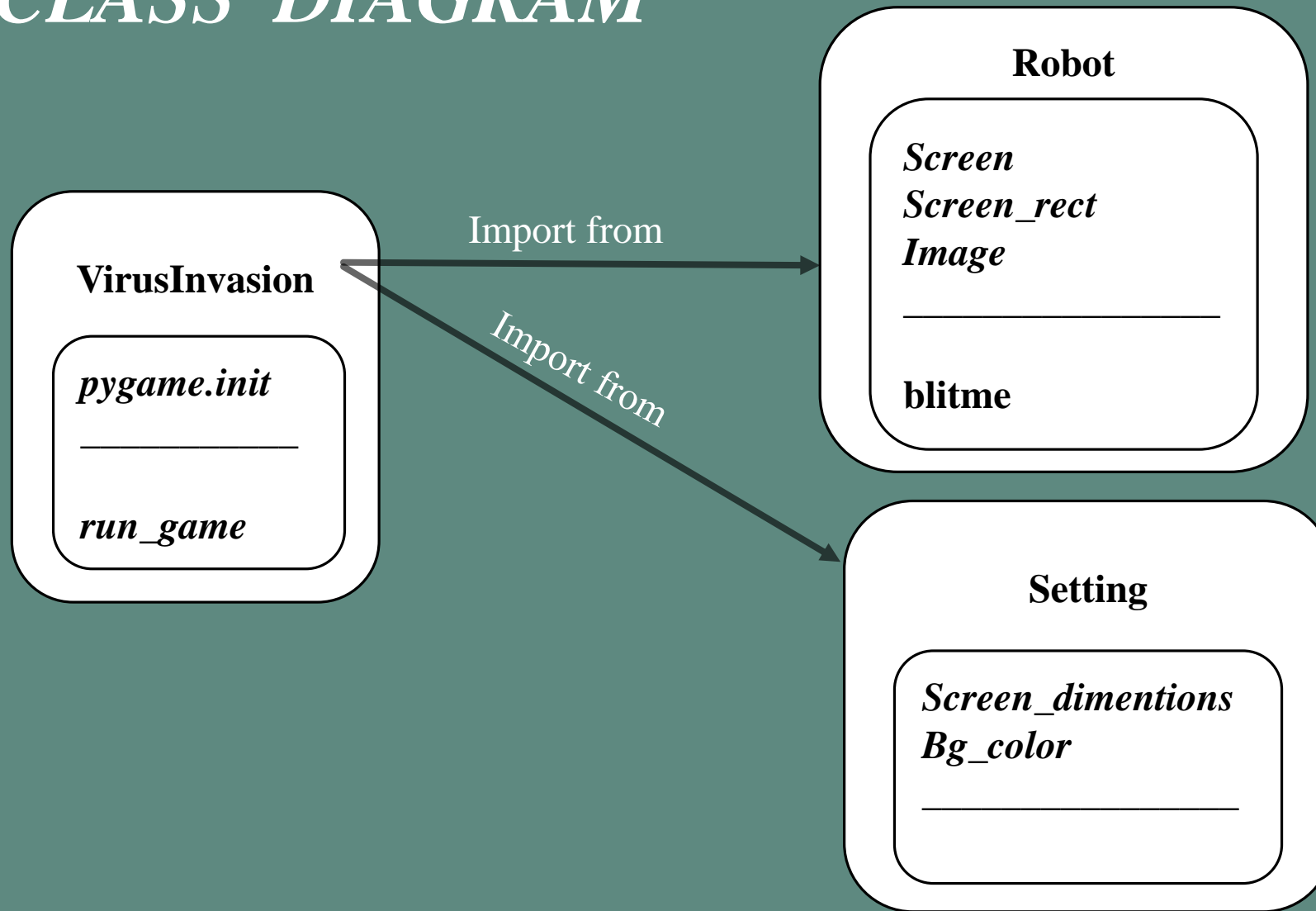
# ***PROJECT MAIN DEVELOPMENT PHASES***



# *FIRST PHASE*



# CLASS DIAGRAM



# CREATING VIRUSINVASION CLASS

We'll make an empty Pygame window by creating a class to represent the game. In your text editor, create a new file and save it as *Virus\_Invasion.py*; then enter the following:

```
import sys
import pygame
class VirusInvasion:
    def __init__(self):      # Class Attributes
        ❶ pygame.init()
        ❷ self.screen = pygame.display.set_mode((1200, 800))
          pygame.display.set_caption("Virus Invasion Game")
    def run_game(self):      # Class Method
        """the main loop for the game."""
        ❸ while True:
            # Watch for keyboard and mouse events.
            ❹ for event in pygame.event.get():
                ❺ if event.type == pygame.QUIT:
                    sys.exit()
            # Make the most recently drawn screen visible.
            ❻ pygame.display.flip()

if __name__ == '__main__':
    # Make a game instance, and run the game.
    vi = VirusInvasion()
    vi.run_game()
```

# CREATING VIRUSINVASION CLASS

## Attributes

- 1 In the `__init__()` method, the `pygame.init()` function initializes the background settings that Pygame needs to work properly
- 2 we call `pygame.display.set_mode()` to create a display window, on which we'll draw all the game's graphical elements. The argument `(1200, 800)` is a tuple that defines the dimensions of the game window, which will be 1200 pixels wide by 800 pixels high. (You can adjust these values depending on your display size.)

### Note:

- ❑ The object we assigned to `self.screen` is called a *surface*. A **surface** in Pygame is a part of the screen where a game element can be displayed.
- ❑ Each element in the game, like a virus or a robot, has its own surface.
- ❑ The **surface** returned by `display.set_mode()` represents the entire game window.
- ❑ When we activate the game's animation loop, this surface will be redrawn on every pass through the loop, so it can be updated with any changes triggered by user input.



# CREATING VIRUSINVASION CLASS

The game is controlled by the `run_game()` Method

- 3 *The while loop contains an event loop and code that manages screen updates.*

An *event* is an action that the user performs while playing the game, such as pressing a key or moving the mouse.

To make our program respond to events, we write this *event loop* to *listen* for events and perform appropriate tasks depending on the kinds of events that occur.

- 4 To access the events that Pygame detects, we'll use the `pygame.event.get()` function.

This function returns a list of events that have taken place since the last time this function was called. Any keyboard or mouse event will cause this for loop to run.

- 5 Inside the loop, we'll write a series of if statements to detect and respond to specific events.

For example, when the player clicks the game window's close button, a `pygame.QUIT` event is detected and we call `sys.exit()` to exit the game

- 6 Tells Pygame to make the most recently drawn screen visible,

When we move the game elements around, `pygame.display.flip()` continually updates the display to show the new positions of game elements and hides the old ones, creating the illusion of smooth movement.

# *CREATING VIRUSINVASION CLASS*

## **create an instance of the game**

At the end of the file, we create an instance of the game, and then call **run\_game()**.

When you run this *Virus\_Invasion.py* file, you should see an empty Pygame window.

## ***Setting the Background Color***

Pygame creates a black screen by default, but that's boring. Let's set a different background color. We'll do this at the end of the `__init__()` method.

Colors in Pygame are specified as RGB colors: a mix of red, green, and blue. Each color value can range from 0 to 255.

# CREATING VIRUSINVASION CLASS

## Setting the Background Color

```
def __init__(self):  
    --snip--  
    pygame.display.set_caption("Virus Invasion Game")  
    # Set the background color.  
    1 self.bg_color = (230, 230, 230)  
  
def run_game(self):  
    --snip--  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            sys.exit()  
    2 # Redraw the screen during each pass through the loop.  
    self.screen.fill(self.bg_color)  
    # Make the most recently drawn screen visible.  
    pygame.display.flip()
```

1 We assign this color to **self.bg\_color**

2 we fill the screen with the background color using the **fill()** method, which acts on a surface and takes only one argument: a color.

# CREATING A SETTINGS CLASS

Instead of adding settings throughout the code, let's write a module called **settings** that contains a class called **Settings** to store all these values in one place. This approach allows us to work with just one settings object any time we need to access an individual setting.

## **settings.py**

```
class Settings:
    """A class to store all settings for Virus Invasion."""
    def __init__(self):
        # Screen settings
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230, 230, 230)
```

# CREATING A SETTINGS CLASS

To make an instance of `Settings` in the project and use it to access our settings, we need to modify `virus_invasion.py` as follows:

**`virus_invasion.py`**

```
--snip--
import pygame
from settings import Settings
class VirusInvasion:
    """Overall class to manage game assets and behavior."""
    def __init__(self):
        """Initialize the game, and create game resources."""
        pygame.init()
        1 self.settings = Settings()
        2 self.screen = pygame.display.set_mode(
            (self.settings.screen_width, self.settings.screen_height))
        pygame.display.set_caption("Virus Invasion Game")
        def run_game(self):
            --snip--
            # Redraw the screen during each pass through the loop.
            3 self.screen.fill(self.settings.bg_color)
            # Make the most recently drawn screen visible.
            pygame.display.flip()
            --snip--
```

1

We import `Settings` into the main program file. Then we create an instance of **`Settings`** and assign it to **`self.settings`**

2

we use the **`screen_width`** and **`screen_height`** attributes of **`self.settings`**,

3

we use **`self.settings`** to access the background color when filling the screen