**Methodological guidelines**

**The Arkanoid game. Part 2**

**🚀STORYLINE:**

The specialists from the ProTeam game development department have taken on a new order to create the game "Arkanoid". The first Arkanoid game was invented in 1986, and it was so popular that it became the standard name for a family of similar games. The ProTeam game designers have come up with their own version of the famous game and handed it over to the developers to implement the functionality.

Continuing the technical task, the developers work with keyboard events and program the automatic movement of the ball sprite.

**⚠️ SUMMARY:**

**The lesson goal** is to implement control of the platform with the keyboard and program the automatic movement of the ball.

In the first half of the lesson, the students plan their work on the project and work with keyboard events. In the second half, they describe the trajectory of the automatic movement of the ball and program it.

**💾 LINKS AND ACCESSORIES:**

* the [presentation](https://docs.google.com/presentation/d/1YFyYf66B0SHM4T8pEqzKac3qCIOgHMS6NI-cSVvwSrU/edit?usp=sharing),
* tasks: [planning](https://backoffice.algoritmika.org/task-preview/33288?level=1&track=1), [implementation](https://backoffice.algoritmika.org/task-preview/33289?track=1&position=1).

# 

# 🎯 EDUCATIONAL OUTCOMES OF THE LESSON

| *After the lesson, the students will:* | *The result will be achieved when the students:* |
| --- | --- |
| * be able to use Pygame commands to create a game template; * be able to create their own class and give it properties and methods; * be able to create a derived class from an existing class; * be able to create instances of their own class and use them to fill lists; * be able to work with lists of objects, using methods for working with lists; * be able to handle keyboard events using Pygame tools; * be able to program the automatic movement of the sprite in the game scene; * be able to work on a project using a mind map and a checklist. | * have participated in the discussions and asked clarifying questions; * can confidently list the basic components of a game in Pygame; * have planned their work on a project using a mind map and a checklist; * have programmed control of the platform with the keyboard and the automatic movement of the ball through the scene. * have answered the teacher’s questions during the review stage. |

**RECOMMENDED LESSON STRUCTURE**

| **Time** | **Stage** | Stage aim |
| --- | --- | --- |
| 10 min | Storyline. Discussion:  Work on the Project | * Highlight the tasks: to program control of the platform and the movement of the ball. * Mark the goal for the current working day on the mind map and add to the checklist with tasks. * Discuss possible difficulties and ways to overcome them. |
| 10 min | Checking qualifications | * Organize confirmation of the developers' qualifications on the topic:   + Handling keyboard events |
| 10 min | Brainstorm:  Controlling the platform with the keyboard | * Discuss the issue of long key presses (in this case, the platform should continue to move). * Formulate an algorithm for controlling the platform using the W and S keys. |
| 15 min | Platform:  Pygame: Arkanoid. Implementation | * Get the students to complete the task "Pygame: Arkanoid. Implementation." |
| 5 min | Break | * Do a warm-up and get the kids to switch to a different activity. |
| 15 min | Brainstorm:  Moving the ball | * Discuss the trajectory of the ball's movement and the parameters that describe it. * Arrive at the idea that we need to limit movement to within the boundaries of the screen. * Introduce the commands for implementing the rebound of the ball off the platform. |
| 25 min | Platform:  Pygame: Arkanoid. Implementation | * Get the students to complete the task "Pygame: Arkanoid. Implementation." |
| 5 min | End of the lesson. Reflection | * Conduct a technical interview on the brainstorming materials. * Suggest that the students study the documentation containing the results of the "brainstorming" sessions. |

**Storyline. Discussion: "Work on the project"**

##### *(10 min)*

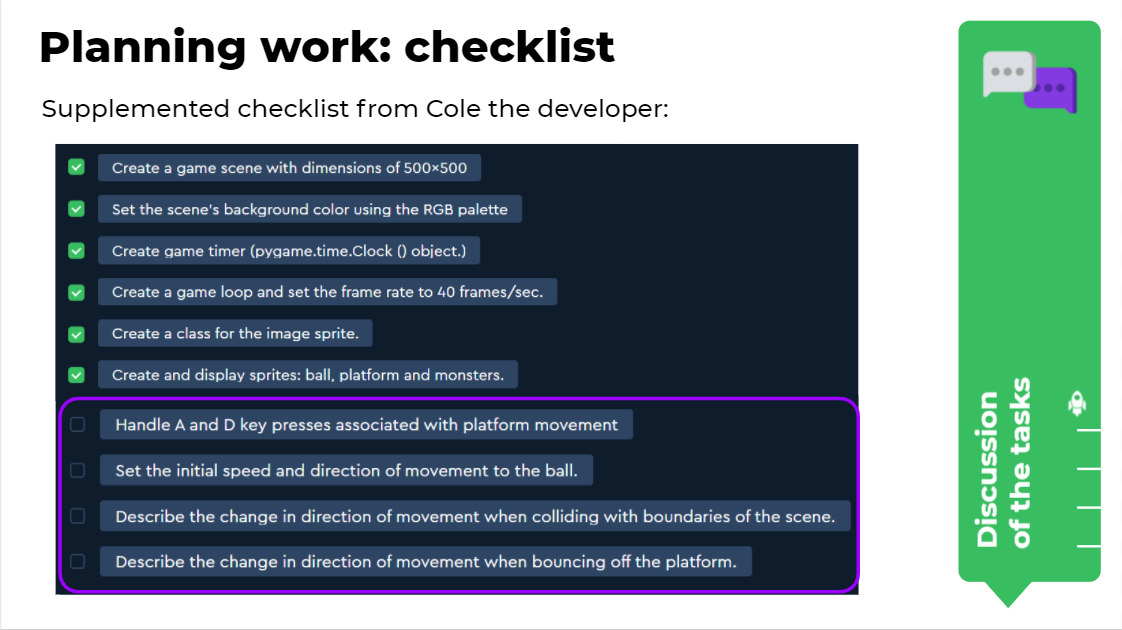
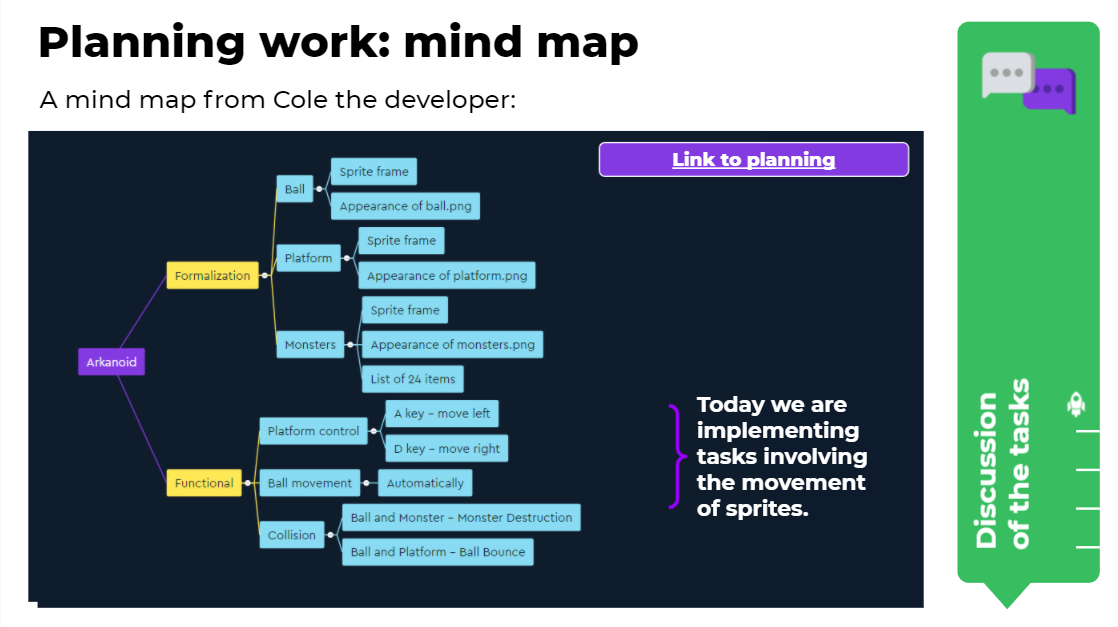
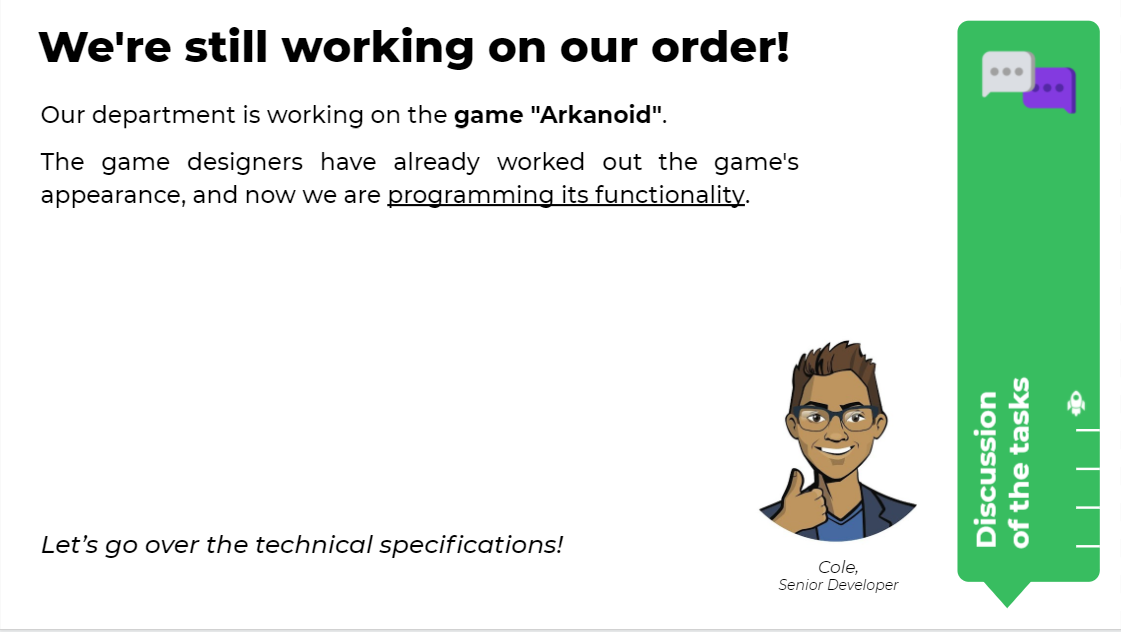
Open the presentation. The developers do not need computers yet.

| *“Hello, colleagues! Today we'll continue working on the development of the Arkanoid game. Last time we planned our work on the project and made a mind map of the game's functionality and a checklist of the tasks. With their help, we created a game template with a background and the main game objects: a ball, a platform, and the monsters.*  *Before continuing our work on the project, we'll make a list of tasks for today and put it in our checklist. Are you ready to work? (the students answer)".* |
| --- |

Remind them about the overall structure of the *project* using a mind map. Highlight the main task — to program the movement of the platform and the ball.

Suggest that they open the checklist and check off the completed tasks. Introduce the new items needed to finalize the game, and add them to the checklist . Then show the version of the updated checklist made by Cole the developer.

While planning the project, draw the students' attention to remembering how to do event handling and the need to learn special commands from Pygame.



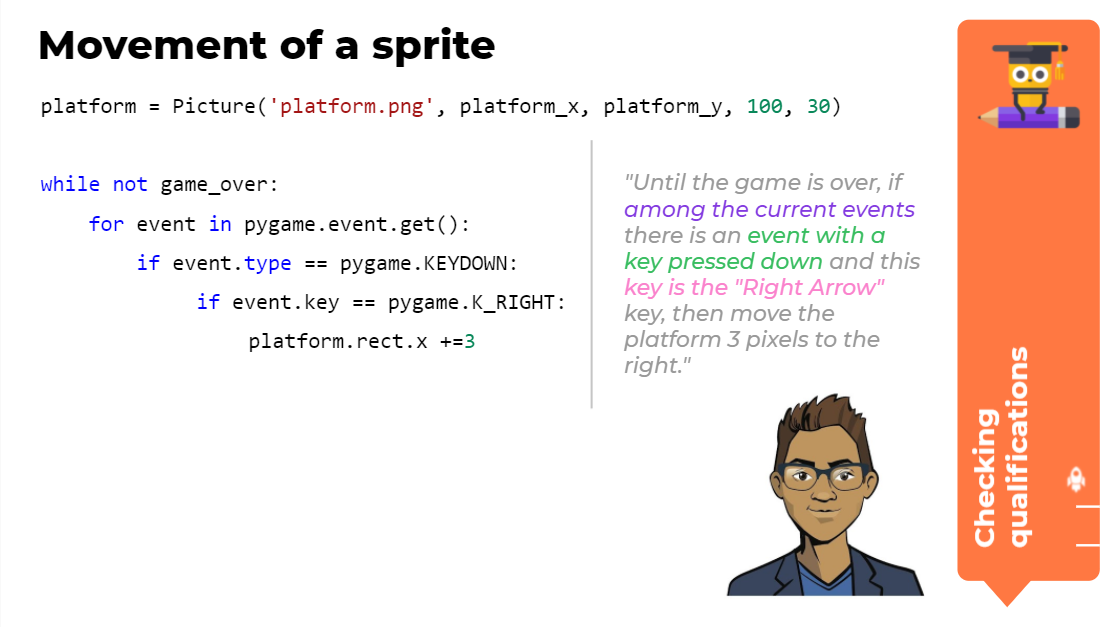
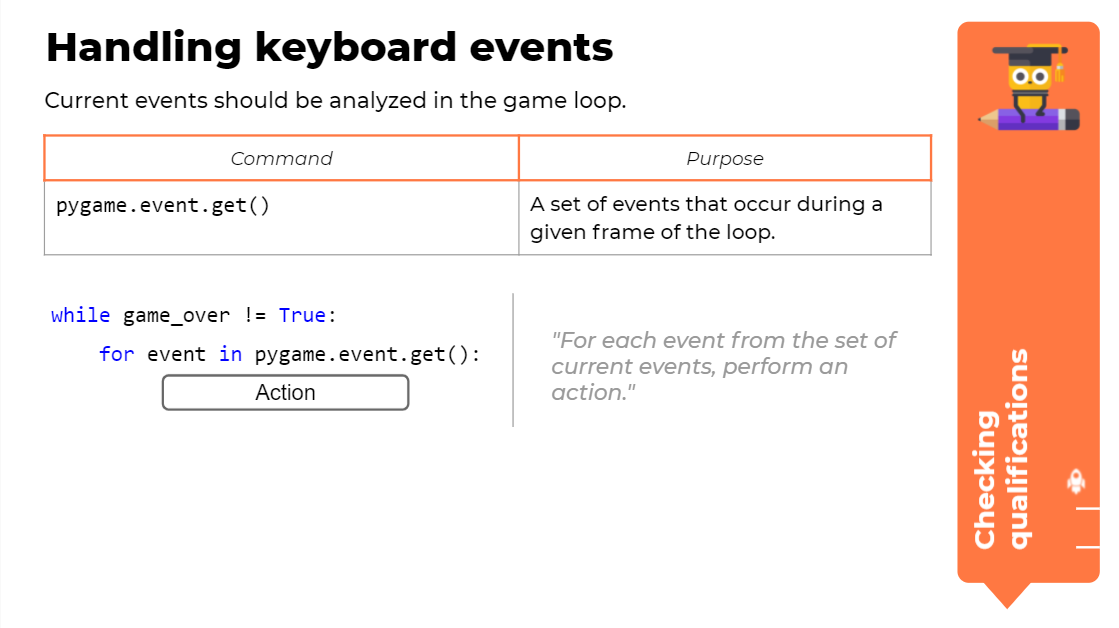
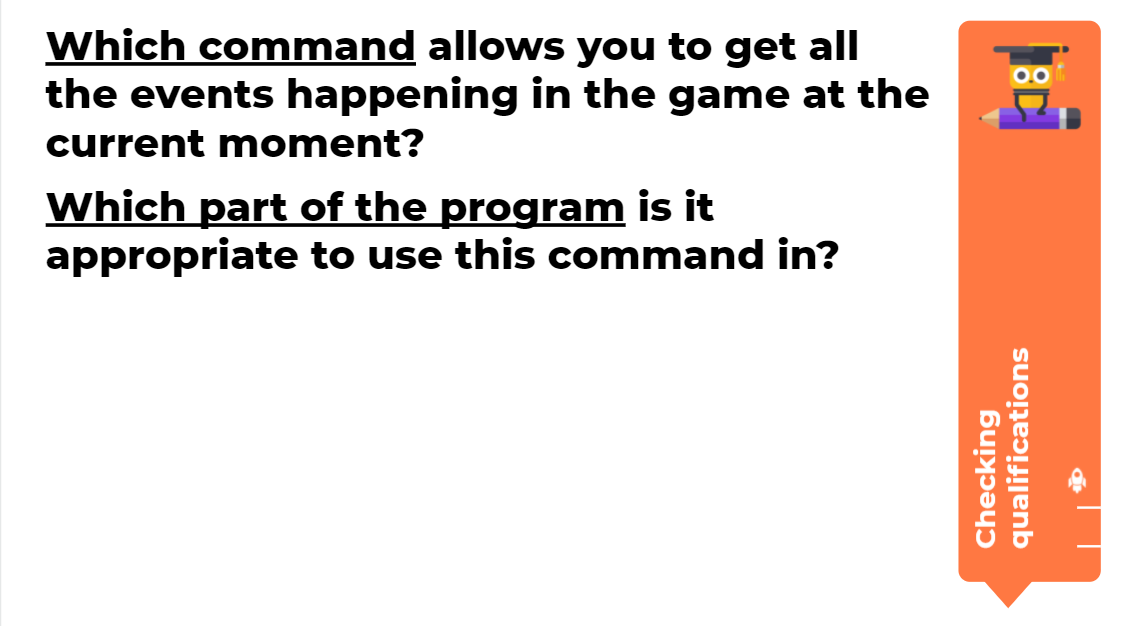
Set the goal for the day and announce what will need to be done.

# 

# Checking qualifications

##### *(10 min)*

Use the presentation to organize confirmation of the developers' qualifications before starting work. This time the procedure passes through the topics: "The basics of event processing" and "Handling keyboard events using Pygame".

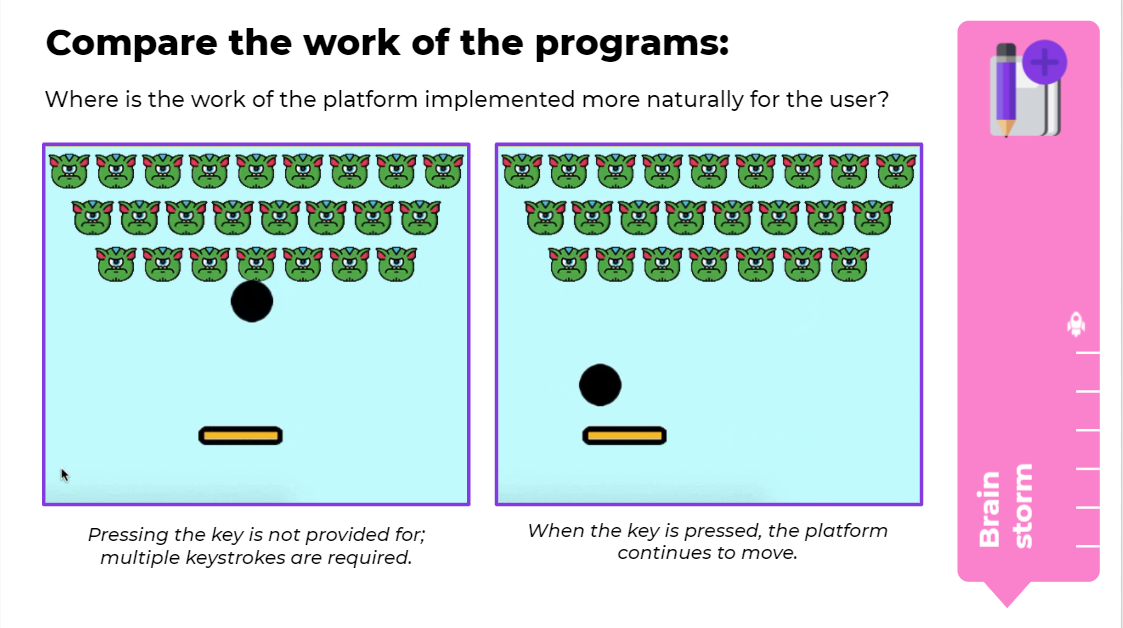
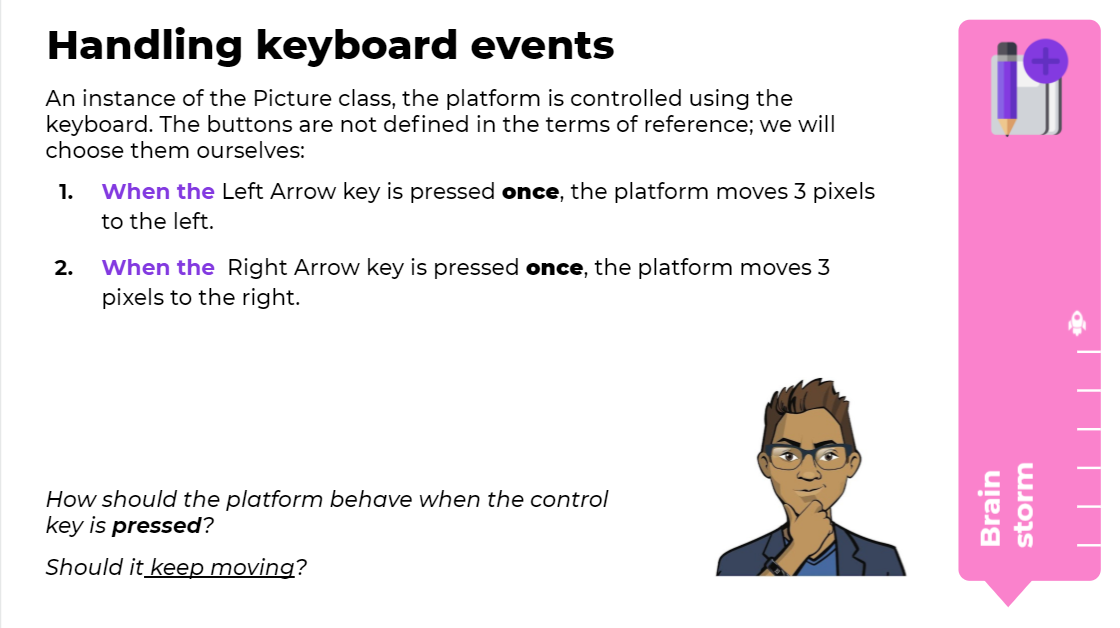


# Brainstorm: Controlling the platform with the keyboard

##### *(10 min)*

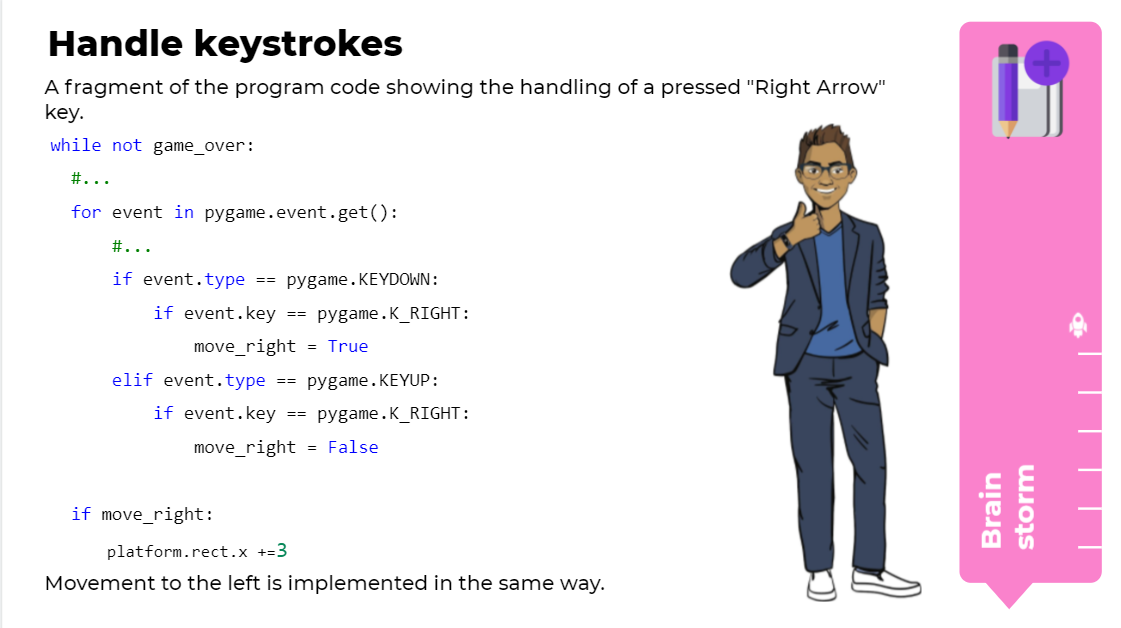
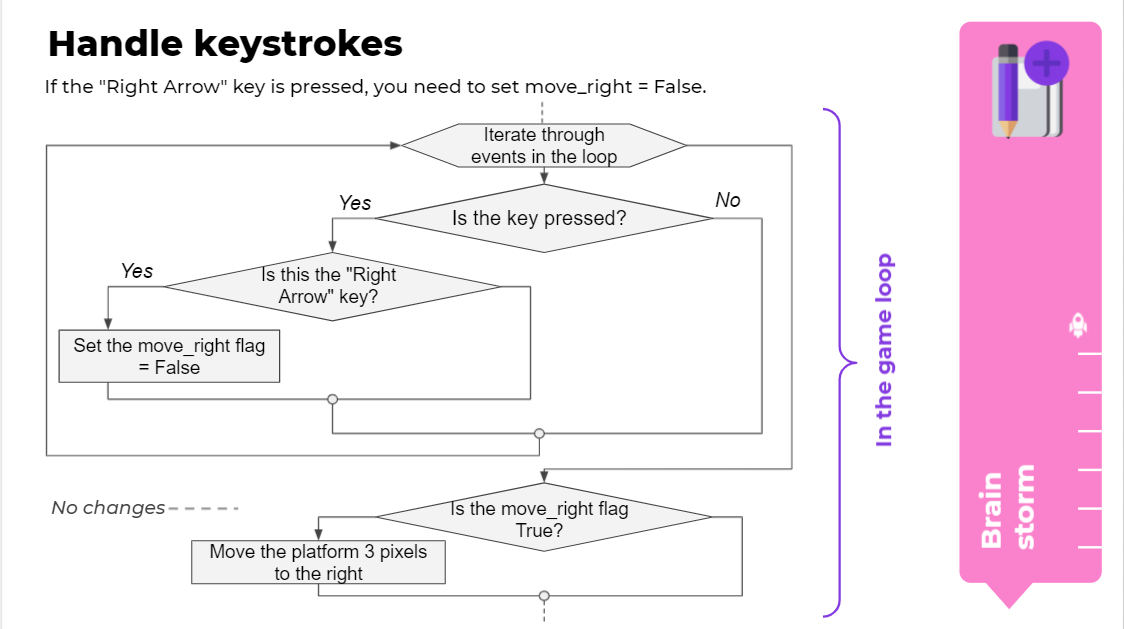
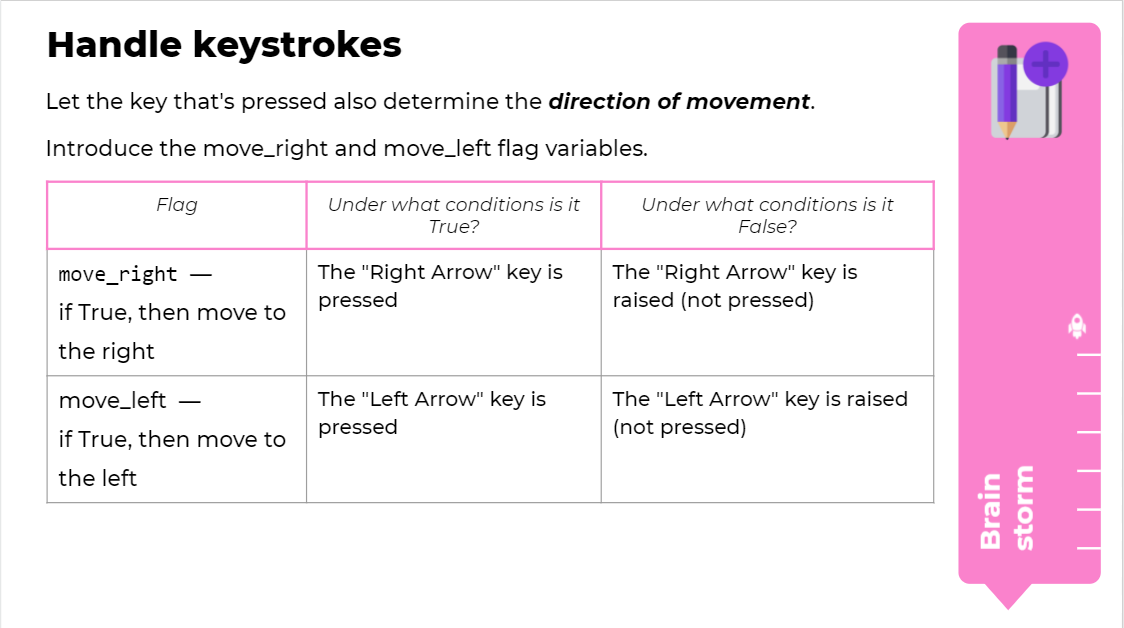
Highlight the task of the first half of the working day — to implement control of the platform using the keyboard. The Platform (an instance of the Picture class) should be controlled using the left and right arrow keys: when the left arrow key is pressed once, the platform moves 3 pixels to the left, and when the right arrow key is pressed, it moves 3 pixels to the right.

Show the slide with a comparison of the two programs and ask the question: "Where is control of the platform implemented more naturally for the user?" Reach the conclusion that the option where holding down the key makes the platform continue to move is more understandable for players, but requires additional discussion.



Let the pressed key determine not just the shift by n pixels, but also the direction of movement as a whole. Enter the move\_right and move\_left flag variables, and use the table to show which conditions the flags take True values under and which they take False under. For example, if the move\_right flag is set to True, the platform moves 3 pixels to the right at each step of the game loop.

**Technical comment**. There are other ways to implement this kind of behavior in the platform. For example, there is another option with a deeper analysis of the events taking place in the "Shooter" project which is done during the second year of study.



Examine the flowchart with a fragment of the algorithm for movement of the platform to the right: If the right arrow key is pressed, you need to set move\_right = True. If the right arrow key is not pressed, you need to set move\_right = False. After a possible change in the values of the flags, a conditional statement is placed responsible for shifting the platform in the right direction.

Discuss the implementation of the now-understood code in the finished program. Sum up the discussion and proceed to working on the platform.

# The platform - Pygame: Arkanoid. Implementation

##### *(15 min)*

##### Organize work on the platform. In the task, the developers need to program the movement of the platform sprite when keys are pressed, and make sure that the platform continues to move when the key is pressed for a long time.

For the developers who have time left over after completing the main task: ask them to think about how they can give the ball sprite an initial speed.

##### The program code is at the end of the guidelines.

# Break

##### *(5 min)*

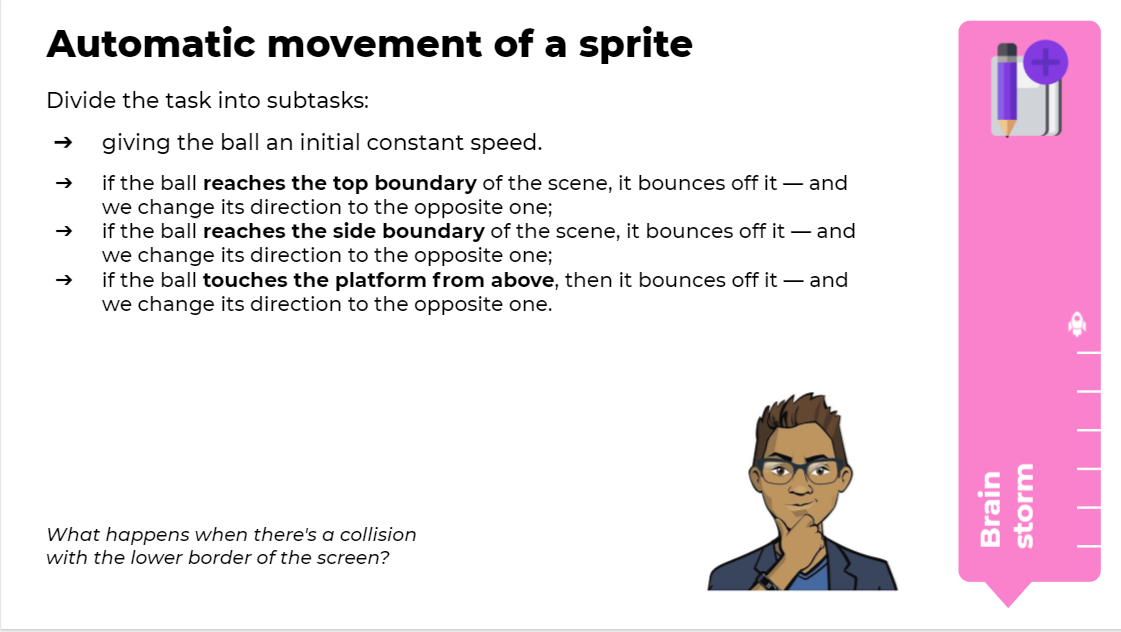
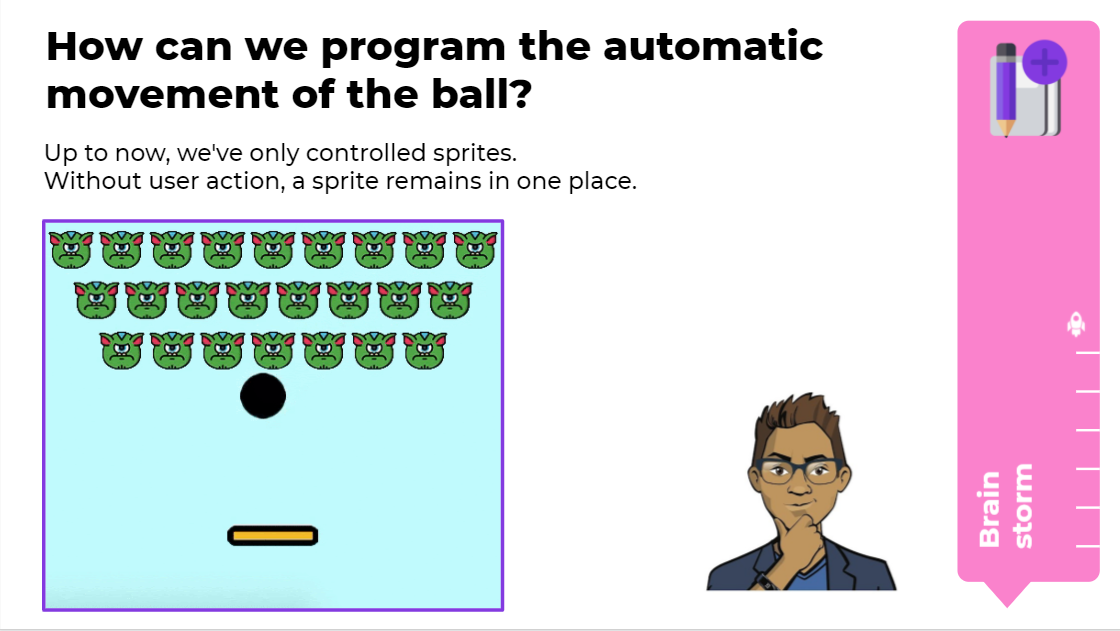
# 

Ask the developers to switch off their computers. The purpose of the break is to shift their focus and do physical activity. Organize one of the [suggested physical activities](https://docs.google.com/document/d/1zRnRvm7tA971qahq6Rjd4nQeRWE5doi2n7gmLKN9g3I/edit).

# Brainstorm: Movement of the ball

##### *(15 min)*

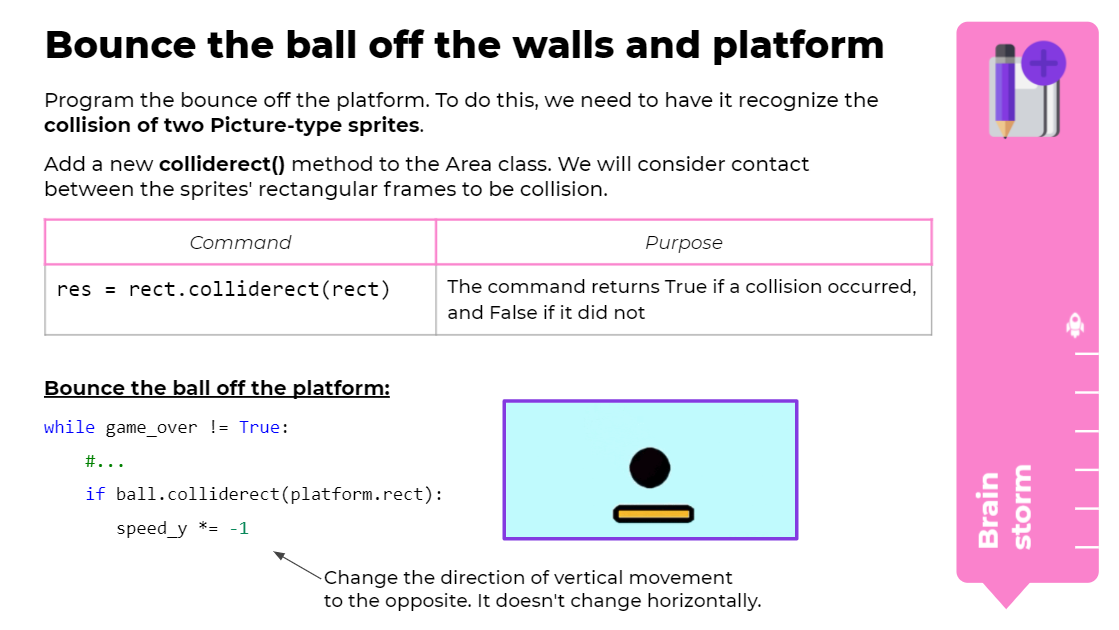
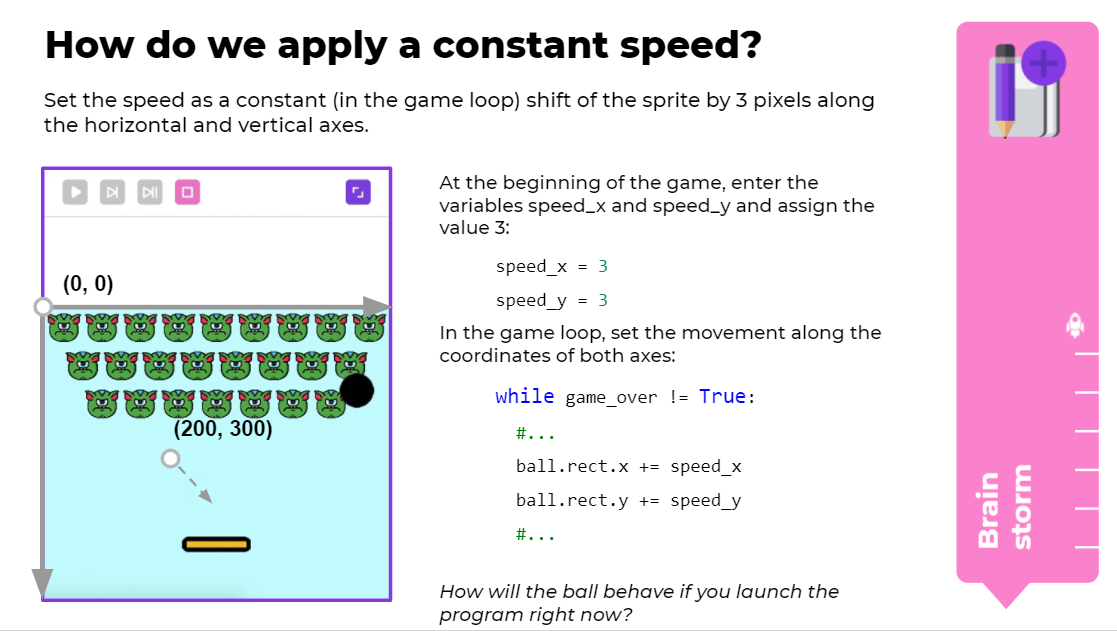
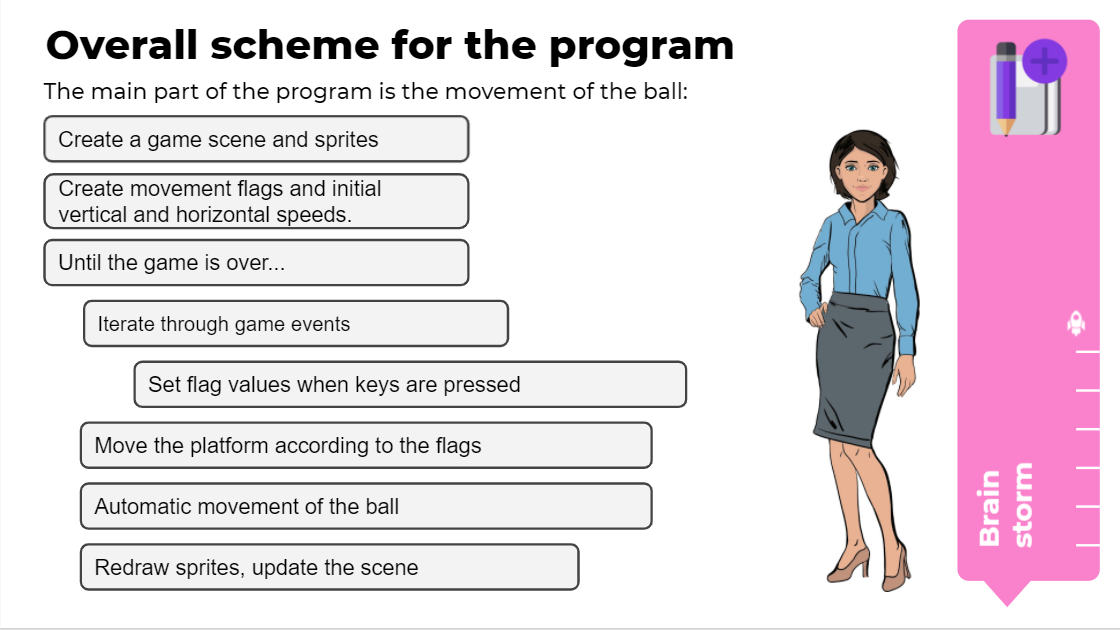
Proceed to implementing the automatic movement of the ball. First we need to understand what restrictions are placed on the ball's movement by the game. Discuss what happens when there is a collision with the top or side border of the scene (the ball bounces off the wall and changes its direction to the opposite one), and what happens when it collides with the lower border of the screen (the lower border of the screen is not an obstacle; when the ball goes over it the player loses and the game ends).



Discuss how we can give the ball a constant speed when the game launches. We need to set the speed as a constant (in the game loop) shift of the sprite by 3 pixels along the horizontal and vertical axes. At the beginning of the game, enter the variables speed\_x and speed\_y and assign them a value of 3: In the game loop, we will set the movement of the ball with an increase in both coordinates.

Proceed to implementing the rebound of the ball off the walls and the platform. To do this, we need to have it recognize the collision of two Picture-type sprites. Add a new colliderect() method to the Area class. We will consider contact between the sprites' rectangular frames to be collision. Discuss the syntax of the new method with the developers.

Discuss the commands for implementing the rebound of the ball off the walls. When a collision occurs, the variables with velocities along the vertical and horizontal axes should change their signs to the opposite ones.

Discuss the implementation of the now-understood code in the finished program. Sum up the discussion and proceed to working on the platform.

# Platform: "Pygame: The Fast Clicker game. Implementation"

##### *(25 min)*

##### Organize work on the platform. In the task, the developers need to give the ball sprite an initial movement speed when the game launches, supplement the Area class with a new method, and program the rebound of the ball off the platform and the walls of the scene.

##### The program code is at the end of the guidelines.

# Wrapping up the lesson

##### *(5 min)*

Draw the developers' attention away from the computers and organize a technical interview based on the materials from the brainstorming sessions. Announce that during the next working day they will finish work on the Arkanoid game and present the results of their work to the customers.

For independent work, provide the developers with the theoretical documentation based on the discussion materials.

**Answers to the tasks:**

**Task "Pygame: Arkanoid. Implementation", part 2.**

import pygame

pygame.init()

back = (200, 255, 255)

mw = pygame.display.set\_mode((500, 500))

mw.fill(back)

clock = pygame.time.Clock()

#variables responsible for platform coordinates

platform\_x = 200

platform\_y = 330

#variables responsible for the ball's direction of movement

dx = 3

dy = 3

#flags responsible for the movement of the platform to the right/left

move\_right = False

move\_left = False

#flag for the end of the game

game\_over = False

#a class from a previous project

class Area():

def \_\_init\_\_(self, x=0, y=0, width=10, height=10, color=None):

'''area: a rectangle in the right place and the right color'''

#memorize the rectangle:

self.rect = pygame.Rect(x, y, width, height)

#fill color - either the passed parameter or the overall background color

self.fill\_color = back

if color:

self.fill\_color = color

def color(self, new\_color):

self.fill\_color = new\_color

def fill(self):

pygame.draw.rect(mw, self.fill\_color, self.rect)

def collidepoint(self, x, y):

return self.rect.collidepoint(x, y)

def colliderect(self, rect):

return self.rect.colliderect(rect)

#class for image objects

class Picture(Area):

def \_\_init\_\_(self, filename, x=0, y=0, width=10, height=10):

Area.\_\_init\_\_(self, x=x, y=y, width=width, height=height, color=None)

self.image = pygame.image.load(filename)

def draw(self):

mw.blit(self.image, (self.rect.x, self.rect.y))

#creating a ball and the platform

ball = Picture('ball.png', 160, 200, 50, 50)

platform = Picture('platform.png', platform\_x, platform\_y, 100, 30)

#creating enemies

start\_x = 5

start\_y = 5

count = 9

monsters = []

for j in range(3):

y = start\_y + (55 \* j)

x = start\_x + (27.5 \* j)

for i in range (count):

d = Picture('enemy.png', x, y, 50, 50)

monsters.append(d)

x = x + 55

count = count - 1

while not game\_over:

ball.fill()

platform.fill()

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_over = True

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_RIGHT: #if the key is pressed

move\_right = True # raising the flag

if event.key == pygame.K\_LEFT:

move\_left = True # raising the flag

elif event.type == pygame.KEYUP:

if event.key == pygame.K\_RIGHT:

move\_right = False #lowering the flag

if event.key == pygame.K\_LEFT:

move\_left = False #lowering the flag

if move\_right: #right movement flag

platform.rect.x +=3

if move\_left: #left movement flag

platform.rect.x -=3

#giving constant acceleration to the ball at x and y

ball.rect.x += dx

ball.rect.y += dy

# if the ball reaches the borders of the screen, change the direction of its movement

if ball.rect.y < 0:

dy \*= -1

if ball.rect.x > 450 or ball.rect.x < 0:

dx \*= -1

# if the ball has touched the racket, change the direction of movement

if ball.rect.colliderect(platform.rect):

dy \*= -1

for m in monsters:

m.draw()

platform.draw()

ball.draw()

pygame.display.update()

clock.tick(40)