Methodological guidelines
Basics of Creating Games

🚀 **STORYLINE:**

The ProTeam company decided to start creating and promoting its commercial products, computer games. They chose the PyGame library as the main tool.

In order to start creating games of different genres, developers need to recall (or study) the basics of working with PyGame.

⚠ **SUMMARY:**

The **lesson goal** is to recall and put into practice the basics of game creation using the PyGame library.

In the first half of the lesson, students recall the concepts of “sprite”, “scene”, “game loop” and apply them to solve the training tasks. In the second half of the lesson, students recall how to handle keyboard events and work on the Catch-up mini-project.

Note. Some students may already be familiar with the PyGame library. However, we recommend starting the module with the basics. Introduce the first lesson to experienced students as a workshop that will help them work on upcoming projects more efficiently.

💾 **LINKS AND ACCESSORIES:**

- ❏ Presentation,
- ❏ Exercises for the lesson: Catch-up (Visual Studio Code).

🎯 EDUCATIONAL OUTCOMES

After the lesson, the students will:	*The result is achieved when the students:*
• be able to list the capabilities of PyGame; • describe how to handle keys using PyGame in their own words; • create the game window, set the background; • create a sprite image, set its properties, and place it on the scene; • create a game loop, configure the frame rate per second; • handle the game completion event (QUIT); • get a list of the keys pressed (key.get_pressed()), extract and handle pressing specific keys.	• have participated in the discussions and asked clarifying questions; • have listed commands for creating and configuring the game window; • have listed commands for creating and configuring the sprite image; • have programmed the game loop to handle keyboard events; • coped with the Catch-up mini-project; • have answered the teacher’s questions during the review stage.

RECOMMENDED LESSON STRUCTURE

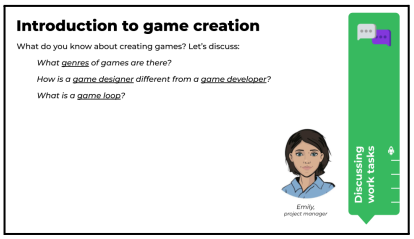
Time	Stage	Stage aims
5 min 	**Storyline. Discussion: Game creation**	❏ Set the storyline-based task: take on projects related to creating games using PyGame. ❏ Arrive at the idea that it is required to review the basics of creating games and implement the Catch-up mini-project.
15 min 	**Brainstorming: Game creation using PyGame**	❏ Discuss the basic concepts associated with creating games: “sprite”, “scene”, “game loop”, FPS. ❏ List methods for creating a game window, sprite image, background. ❏ Go over the game completion event. ❏ Analyze the task of creating a play space with two sprites.
20 min 	**Platform: “VSC. PyGame: Catch-up”**	❏ Organize completion of tasks 1 and 2 of the “PyGame: Catch-up” exercises.
5 min 	**Break**	❏ Do a warm-up or change students’ activity.
15 min 	**Brainstorming: Handling Keyboard Events**	❏ Discuss that when objects change their position it is required to refresh the screen at a rate that is comfortable for the eye. ❏ Discuss handling keyboard events using the keys.pressed() method. ❏ Supplement the Catch-up program with the arrow keys press handling to move the sprites.
25 min 	**Platform: “VSC. PyGame: Catch-up”**	❏ Have the students complete task 3 of the “PyGame: Catch-up” exercises.
5 min 	**Wrapping up the lesson. Reflection**	❏ Conduct a technical interview based on the brainstorming material. ❏ Suggest that the students complete the extra exercise on the VS Code for additional practice.

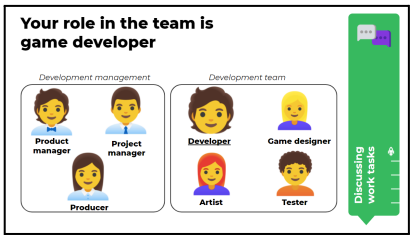
Storyline. Discussion: Game creation

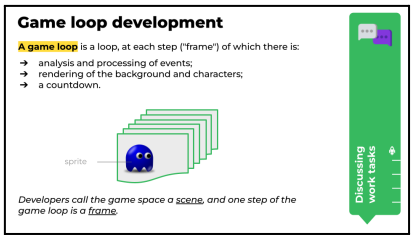
(5 min.)

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

> *"Hello, colleagues! The ProTeam company decided to start creating and promoting its commercial products, computer games. They chose the PyGame library as the main tool".*







Discuss what the developers know about games and how they are made: what games they play on their home computers. What genres of games do they know? Do they know how a game designer differs from a game developer?

Mention that in their work they will be focusing on game development and setting up the game loop. Briefly explain that a game loop is a loop that has events handled at every step (in every "frame") (events of the external world and the game itself, characters and scenes are being drawn, time is counted).

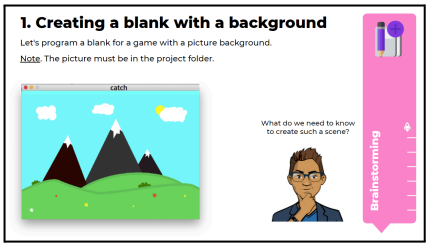
Show the slide with the expected interface of the interactive Maze game. Tell the developers that to create such a game, they need to be comfortable with the basics of PyGame. With this end in view, they will implement the training Catch-up project during the current workday.

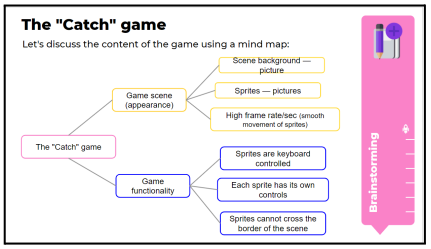
Set the goal for the day and announce what shall be done.

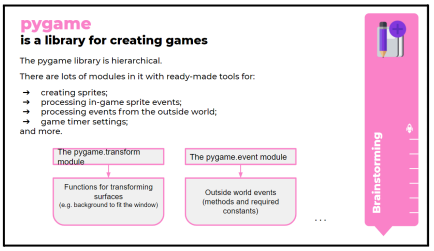
Brainstorming: Game creation using PyGame

(15 min)

Invite the developers to study the appearance of the Catch-up game and list the app's functionality elements. Use a mind map to discuss the content of the game in detail. For example, the "Scene" (external representation of the game) and "Functionality" can become the basic components of your mind map. Think about which components of the PyGame library should you study first?





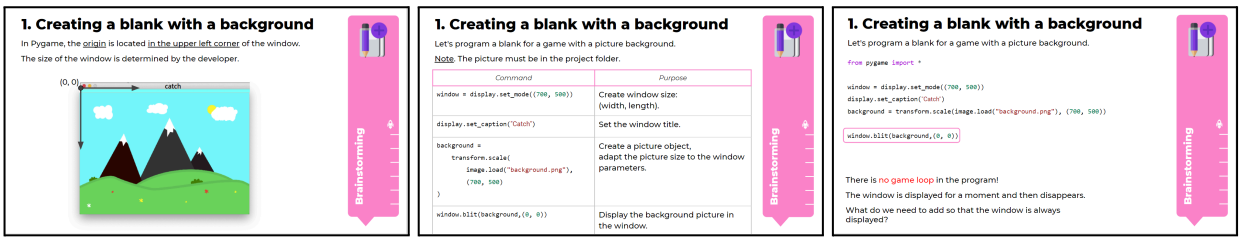


Move on to analyzing the content of pygame. Tell the kids that like many other libraries, pygame is hierarchical and consists of many modules. Give examples of such modules: transform and event.

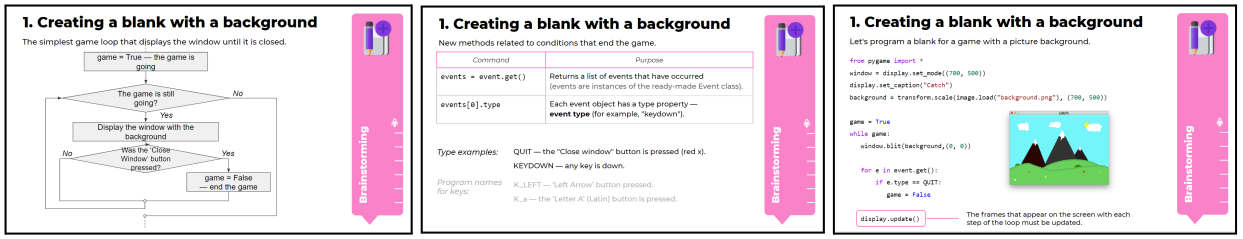
Discuss with the developers their first step in creating the Catch-up game: creating a blank with a background. Remind that the images used must be in the project folder.

> *"In Pygame, the origin is located in the upper left corner of the window. The size of the window is determined by the developer".*

Demonstrate the developers methods for creating a game window and a picture background. Mention that they can resize the picture to fit the window. Emphasize that even the simplest code creating the game window and setting the background will not work correctly without a game loop (the window will open and close immediately)!



Arrive at the idea that it is required to program the simplest game loop displaying the window with a background until the user clicks on the "Close Window" button (red cross). Discuss the syntax of the required methods from the event module of the PyGame library. Specify that frames appearing on the screen with each step of the loop must be updated using the *display.update()* command.



Say that sprites in the form of images (e.g., game characters) can also be adapted to the size we want (e.g., 100×100 pixels) and positioned at an arbitrary point in the window using coordinates.

Combine the fragments analyzed into a program, sum up your discussion, and proceed to work on the VS Code.

Platform: “VSC. PyGame: Catch-up”

(20 min)

Arrange the work on the VS Code. Invite the students to complete tasks 1 and 2 in the “PyGame: Catch-up” exercise.

You will find answers to the exercises at the end of the methodological guidelines.

Break

(5 min.)

Ask the developers to switch off their computers. The purpose of the break is to shift attention and have a warm up.

Brainstorm: Handling Keyboard Events

(15 min)

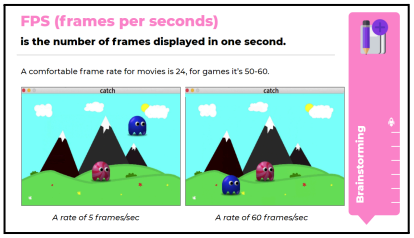
Move on to the discussion of the next step in creating a game: exercise the FPS and sprites control using the keyboard. Show the slide comparing how the two games work.

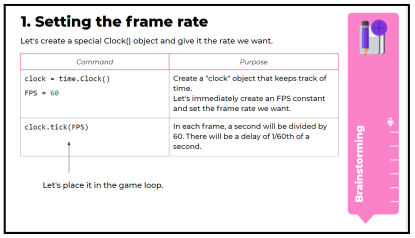
> *“It is noteworthy that sprites move at the same speed. But why does the sprite in the game on the right move more smoothly and quickly? (Students’ answers). This actually happens because the game frames refresh with a different frame per second rate”.*

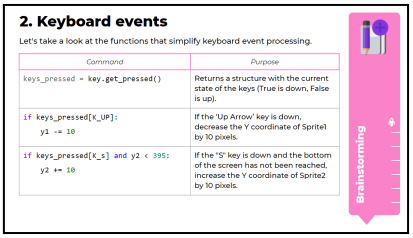
Tell the developers that every game has an important characteristic: the number of frames displayed per second. They also use the abbreviation FPS (frames per second). All other things being equal, a high refresh rate looks better but requires higher computer performance. For games, the frame rate comfortable for human eyes is 50–60 frames/sec.

To set the frame rate, we need to introduce a new object, Clock(), which is a “clock” that keeps track of time. The FPS we want is set in the game loop using the tick() method.

Technical comment. Actually, the tick() method does not set specifically the FPS. In fact, the second will be divided by 60 in each frame, and the next frame will appear with a delay of 1/60 of a second.



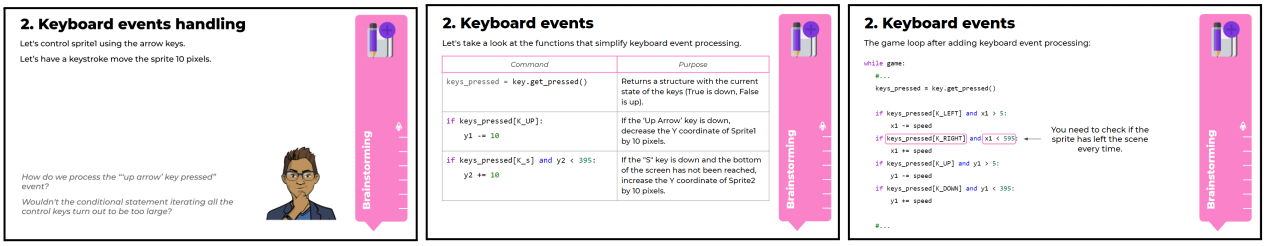




Go to the discussion of moving the sprite using the keyboard. It turns out that handling all possible events “point-blank” will be too cumbersome. It will be especially difficult to program a situation where you hold down an arrow key and the sprite continues to move.

Show the syntax of the get_pressed() method returning the set of keys pressed. Then handling of the keyboard events will be reduced to the following analysis: is there a certain key in the structure received in this frame or not.

Technical comment. In fact, the keys.get_pressed() command returns a sequence of zeros and ones. At the same time, every number always corresponds to the same key, for example, the “right arrow” is 275. But there is no need to remember these numbers by heart, since the keys can be accessed by constants, for example, K_UP.



Mention that movement of the sprite should be limited to the size of the screen. Wrap up the discussion and begin working on the VS Code.

Platform: “VSC. PyGame: Catch-up”

(25 min)

Arrange the work on the VS Code. Invite the students to complete task 3 in the “PyGame: Catch-up” exercise.

End of the lesson

(5 min.)

Use the presentation to sum up the workday. Conduct a technical interview with questions about the brainstorming materials.

Exercises answers

Exercise “VSC. PyGame: Catch-up”.

```
from pygame import *

window = display.set_mode((700, 500))
display.set_caption("catch")
background = transform.scale(image.load("background.png"), (700, 500))

#parameters of the image sprite
x1 = 100
y1 = 300

x2 = 300
y2 = 300

sprite1 = transform.scale(image.load('sprite1.png'), (100, 100))
sprite2 = transform.scale(image.load('sprite2.png'), (100, 100))
speed = 10

#game loop
run = True
clock = time.Clock()
FPS = 60

while run:
   window.blit(background,(0, 0))
   window.blit(sprite1, (x1, y1))
   window.blit(sprite2, (x2, y2))

   for e in event.get():
       if e.type == QUIT:
           run = False

   keys_pressed = key.get_pressed()

   if keys_pressed[K_LEFT] and x1 > 5:
       x1 -= speed
   if keys_pressed[K_RIGHT] and x1 < 595:
       x1 += speed
   if keys_pressed[K_UP] and y1 > 5:
       y1 -= speed
   if keys_pressed[K_DOWN] and y1 < 395:
       y1 += speed

   if keys_pressed[K_a] and x2 > 5:
       x2 -= speed
   if keys_pressed[K_d] and x2 < 595:
       x2 += speed
   if keys_pressed[K_w] and y2 > 5:
       y2 -= speed
   if keys_pressed[K_s] and y2 < 395:
       y2 += speed

   display.update()
   clock.tick(FPS)
```