Module 6. Lesson 8. The game "Arkanoid". Part 3

Checking qualifications

Demonstrate your knowledge of: methods for working with text objects!

Checking qualifications

How do we create a font object ?

Name the command to create the font <u>Verdana</u>, size <u>70</u>.

Checking qualifications

Working with text

Command	*Purpose*
`font1 = pygame.font.Font(None, 70)`	Set the font / Create a font object with the parameters: font — default, size — 70.
`font2 = pygame.font.SysFont('verdana', 70)`	Set the font / Create a font object with the parameters: font — Verdana, size — 70.

In the online browser environment, we will use this command in particular

Checking qualifications

How do we **create** and **display text** using a given font (without the Label class)?

Working with text

Command	*Purpose*
`font1 = pygame.font.Font(None, 70)`	Set the font / Create a font object with the parameters: font — default, size — 70.
`font2 = pygame.font.SysFont('verdana', 70)`	Set the font / Create a font object with the parameters: font — Verdana, size — 70.

```
image = pygame.font.SysFont('verdana', fsize).render(text, True, text_color)
```

$\downarrow$

"Create and display the label 'text' with the color 'text_color', Verdana font and size 'fsize'".

Checking qualifications

How do we **create** and **display text** as an instance of the Label class?

If you find it difficult to answer, then look at the code from the Fast Clicker project.

Checking qualifications

Text as an instance of the Label class

First you need to create a text object and then set the label.

```
text = Label(150, 150, 50, 50, color)
```

The constructor of the Area class is called.

Coordinates

Width and height

Color

```
text.set_text('YOU LOSE', 60, (255,0,0))
```

The method of the Label class.

Size (font size)

Checking qualifications

Qualifications confirmed!

Great, you are ready to brainstorm and complete your work task!

Brainstorm:

Winning and losing

Winning and losing conditions

To complete the game, we just need to supplement the game loop with winning and losing conditions.

Situation	*Trigger condition*
The player wins	All monster sprites have been destroyed (the ball touched every monster)
The player loses	The ball goes below the platform before victory is achieved (it doesn't bounce off the lower boundary, so there is no chance to win)

Brain storm

Winning and losing conditions

To complete the game, we just need to supplement the game loop with winning and losing conditions.

Situation	*Trigger condition*
The player wins	All monster sprites have been destroyed (the ball touched every monster)
The player loses	The ball goes below the platform before victory is achieved (it doesn't bounce off the lower boundary, so there is no chance to win)

Each of these situations means the *end of the game* (the text is displayed, the game ends).

Brain storm

Winning and losing conditions

To complete the game, we just need to supplement the game loop with winning and losing conditions.

Situation	*Trigger condition*
The player wins	All monster sprites have been destroyed (the ball touched every monster)
The player loses	The ball goes below the platform before victory is achieved (it doesn't bounce off the lower boundary, so there is no chance to win)

The analysis of the conditions occurs at each step of the game loop.

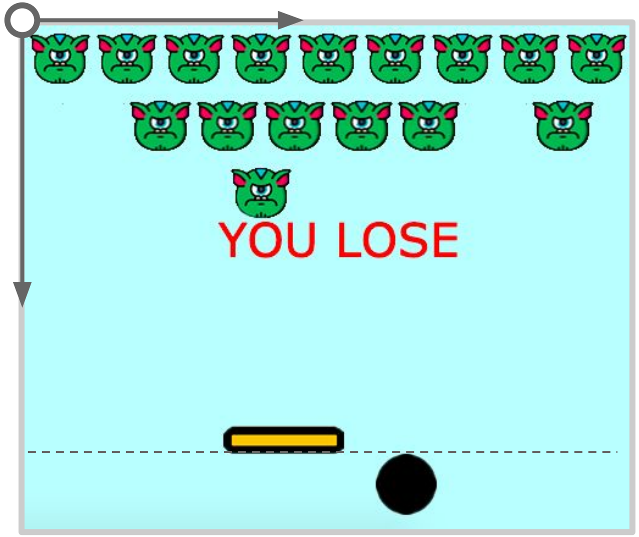
Brain storm

Player loses

The loss occurs if the ball goes below the platform.

For this, the Y coordinate of the platform (which is constant) must be compared with the current Y coordinate of the ball.

```
#...
while not game_over:
  #...
  if ball.rect.y > (platform_y + 20):
      time_text = Label(150, 150, 50, 50, back)
      time_text.set_text('YOU LOSE', 60, (255,0,0))
      time_text.draw(10, 10)
      game_over = True
```



Player loses

The loss occurs if the ball goes below the platform.

For this, the Y coordinate of the platform (which is constant) must be compared with the current Y coordinate of the ball.

```
#...
while not game_over:
  #...
  if ball.rect.y > (platform_y + 20):
      time_text = Label(150, 150, 50, 50, back)
      time_text.set_text('YOU LOSE',60, (255,0,0))
      time_text.draw(10, 10)
      game_over = True
```

Until the game is over...

If the ball went below the platform...

Display the text for losing and switch the "game over" flag to True.

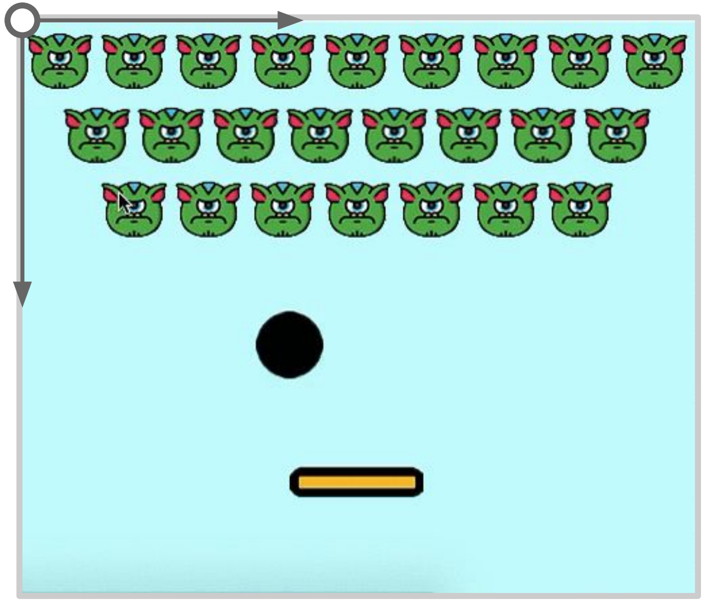
Brain storm

Destruction of all the monsters means the player wins

They win if all the monsters are destroyed.

A monster is destroyed if the ball touches it (it disappears from the screen and is removed from the monsters list).

```
#...
while not game_over:
  #...
  for m in monsters:
     m.draw()
     if m.rect.colliderect(ball.rect):
         monsters.remove(m)
         m.fill()
         dy *= -1
```



Brain storm

Destruction of all the monsters means the player wins

They win if all the monsters are destroyed.

A monster is destroyed if the ball touches it (it disappears from the screen and is removed from the monsters list).

```
#...
while not game_over:
  #...
  for m in monsters:
      m.draw()
      if m.rect.colliderect(ball.rect):
          monsters.remove(m)
          m.fill()
          dy *= -1
```

Until the game is over...

For each monster from the list of monsters...

If the ball has touched a monster, then <u>remove the monster</u> from the list and change the direction of the ball's movement.

Brain storm

The player wins

They win if all the monsters are destroyed.

A monster is destroyed if the ball touches it (it disappears from the screen and is removed from the monsters list).

```
#...
  if len(monsters) == 0:
      time_text = Label(150, 150, 50, 50, back)
      time_text.set_text('YOU WIN',60, (0,2 00, 0))
      time_text.draw(10, 10)
      game_over = True
```

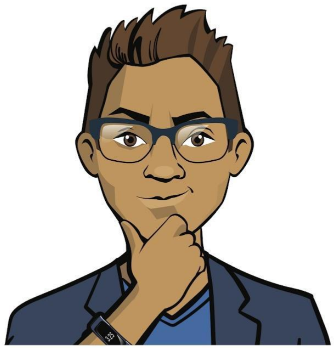
If there are no monsters left...

Display the text for winning and switch the "game over" flag to True.

Brain storm

Your tasks:

1. Program the **losing condition** (the ball goes below the platform) and its accompanying text.
2. Program the **victory condition** (all monsters are destroyed) and its accompanying text.



Brain storm

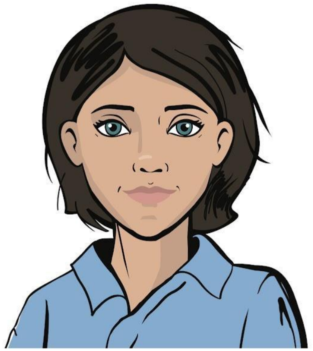
Brainstorm:

Project presentation

Project presentation

You already know that work on an order doesn't end when the program is complete.

Today we have to present the product to the customer. A ***presentation*** will help us with this.



Brain storm

Project presentation

The basic structure includes 3 components. The second step can be expanded upon.

Step 1 You talk **about the goal of the project.**

(What needed to be programmed?)

Step 2 You explain the **solution to the project tasks**.

(How was it programmed?)

Step 3 You reveal **the future prospects for the project.**

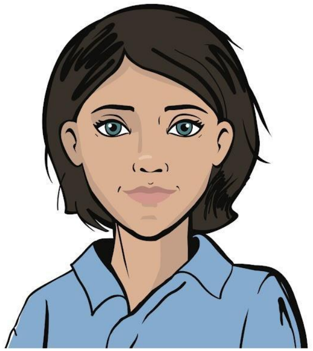
(How can we improve/refine the project?)

Brain storm

Project presentation

A more detailed diagram of the game presentation:

1. "We were given a **task** to program...".
2. "The game is made up of the following **components**...".
3. "A distinctive **feature** of the game is...",
4. "Let's move on to **demonstrating** the program ...".
5. "In the future, the project **can be refined** ..."
6. "Thank you for your attention! I will now take any questions you have".



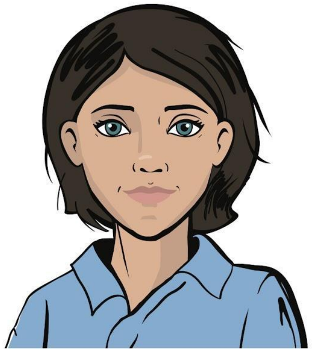
Brain storm

How can we make it visual?

Any story becomes more interesting if the speaker uses a presentation or other visual material.

We worked on the same project, so we will **make a joint report**.

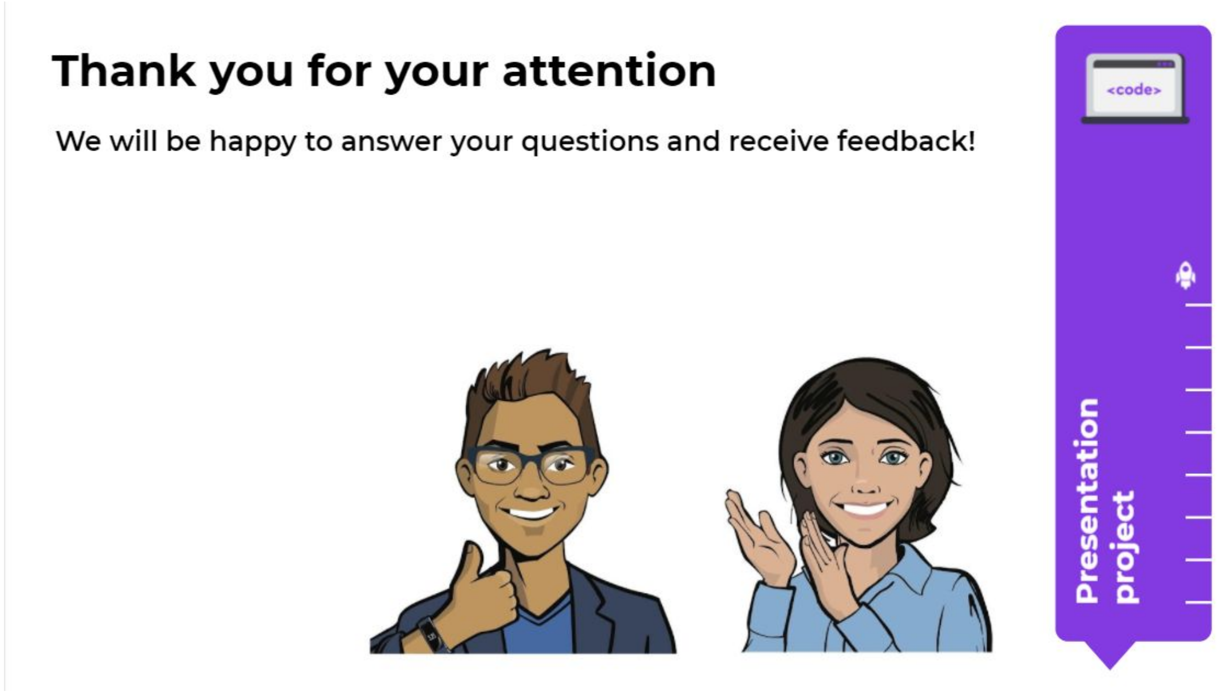
The reference points of the report and illustrations will be plac on slides using Google Slides.



Brain storm

Google Slides: Getting started

Let's get acquainted with the basics of working with presentations in the cloud!
To get started, all you need is a browser and an internet connection.



Thank you for your attention

We will be happy to answer your questions and receive feedback!

<code>
Presentation project

Google Slides: Getting started

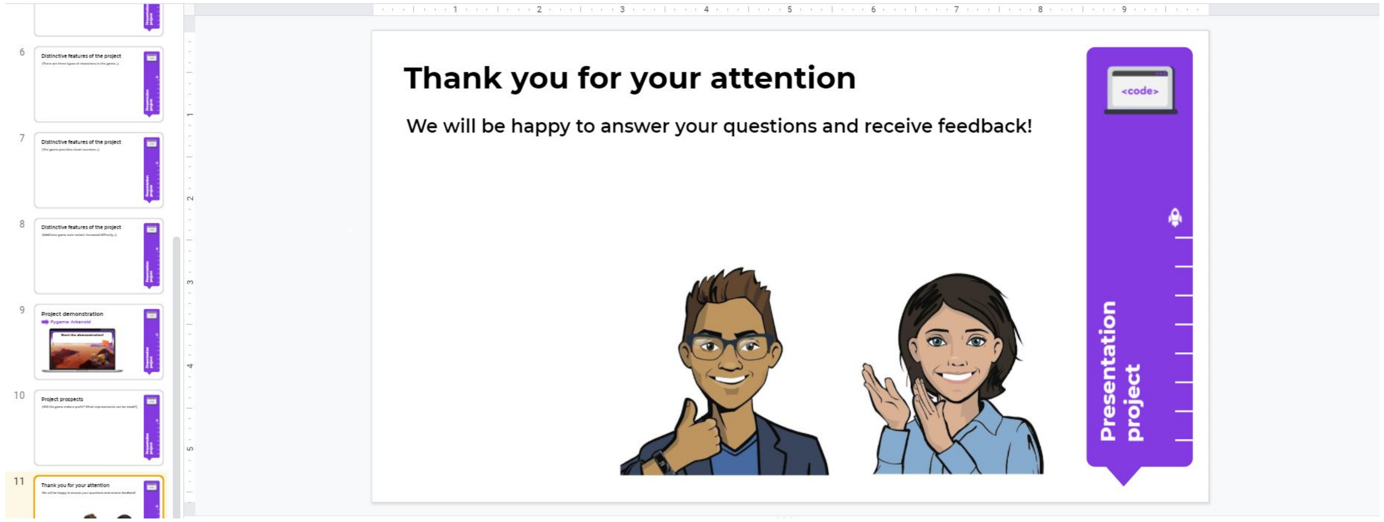
Be careful! You will work in one **shared** document.
Do not delete any slides or their content made by your colleagues!

We will distribute areas of responsibility, so stick to them in your work.

Brain storm

Google Slides: Getting started

You need to register to **create** your own presentation.
You don't need your own account to **edit** someone's document.



6 Distinctive features of the project

7 Distinctive features of the project

8 Distinctive features of the project

9 Project demonstration

Pygame: Arkanoid

Start the demonstration!

10 Project prospects

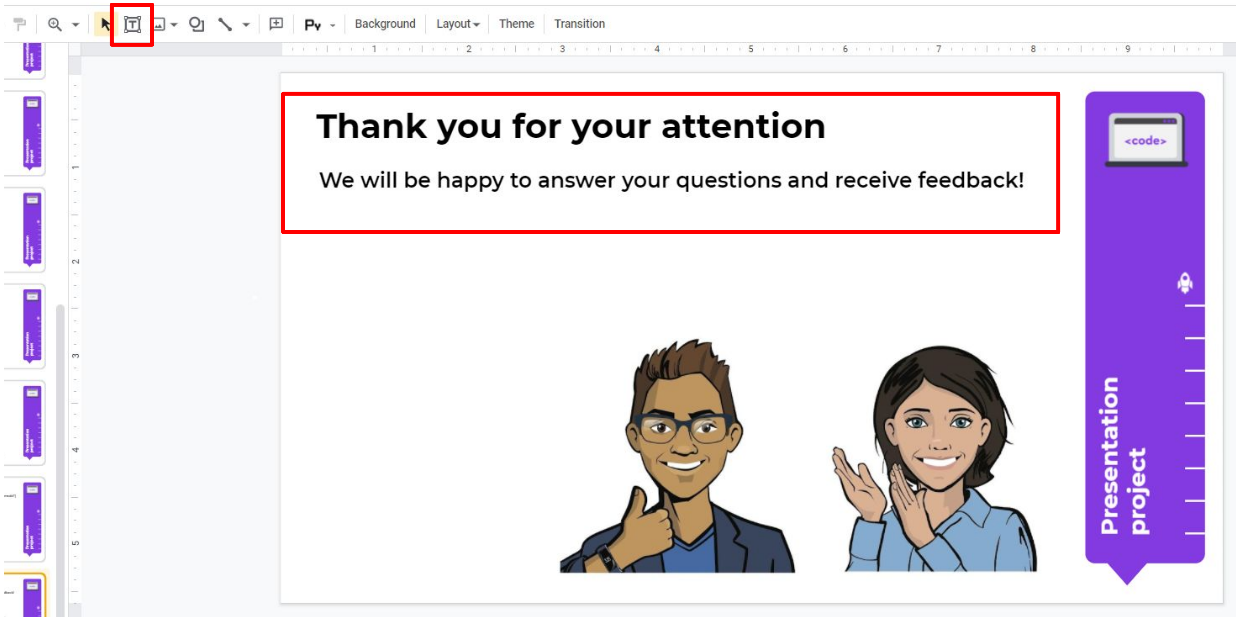
11 Thank you for your attention

Thank you for your attention

We will be happy to answer your questions and receive feedback!

Google Slides: Getting started

Click on the icon to create your own text object.
Click on the available text to edit it.

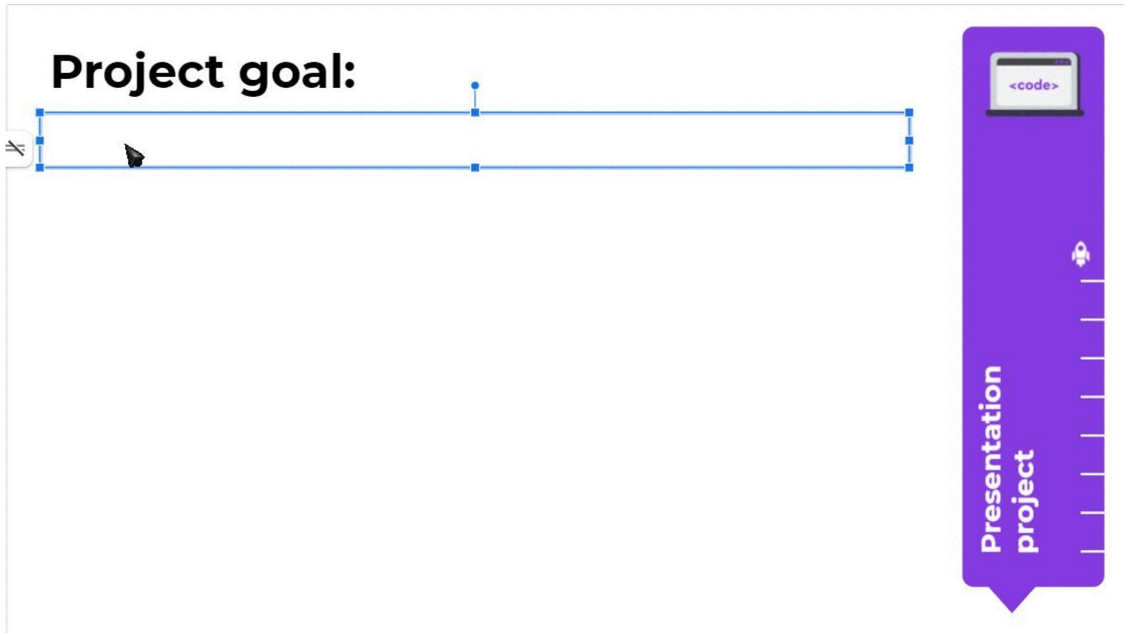


Thank you for your attention

We will be happy to answer your questions and receive feedback!

Google Slides: Getting started

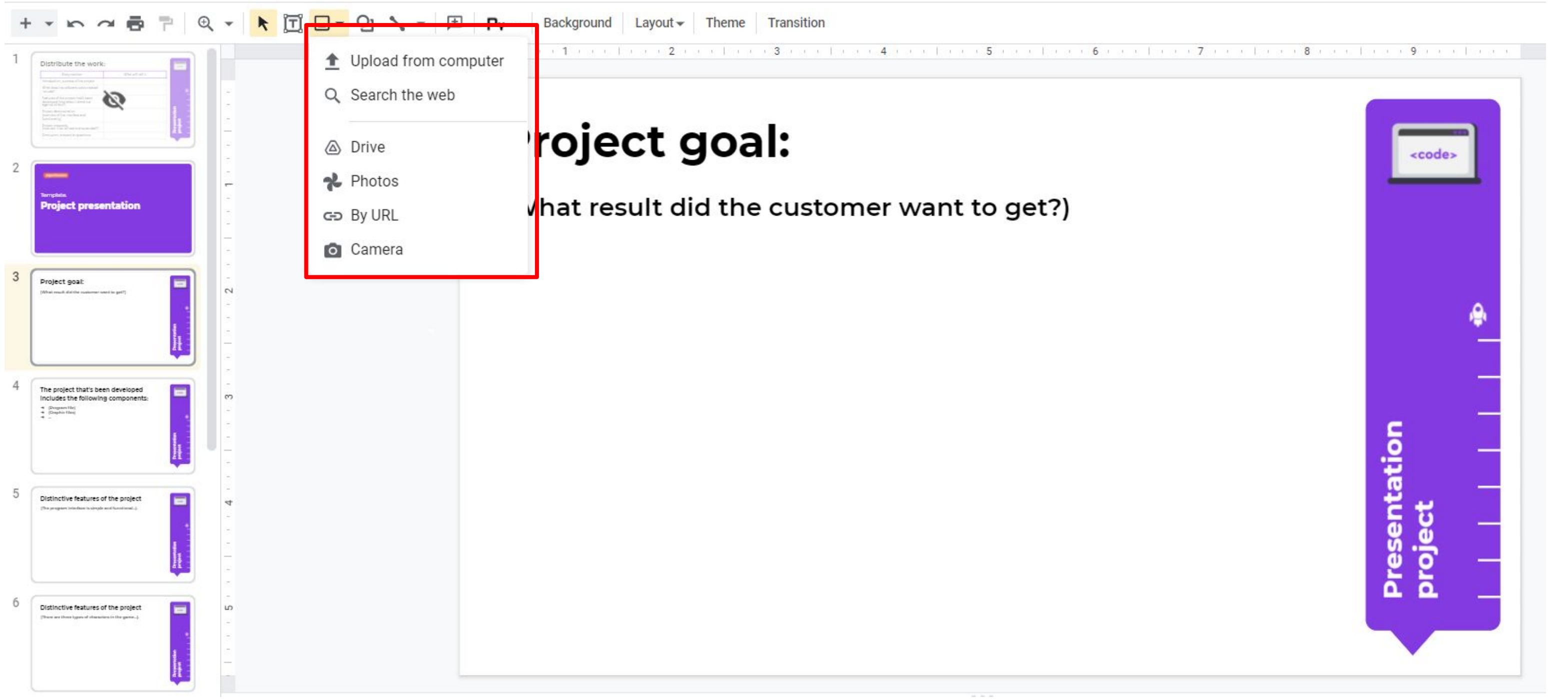
The font and its parameters can be edited:

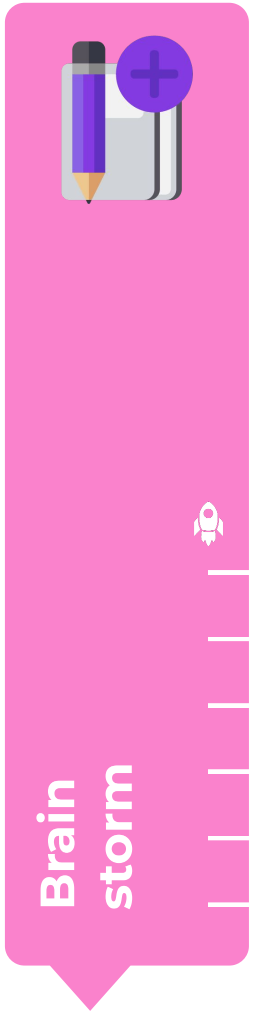


Google Slides: Getting started

Click on the icon to add an image from your computer.

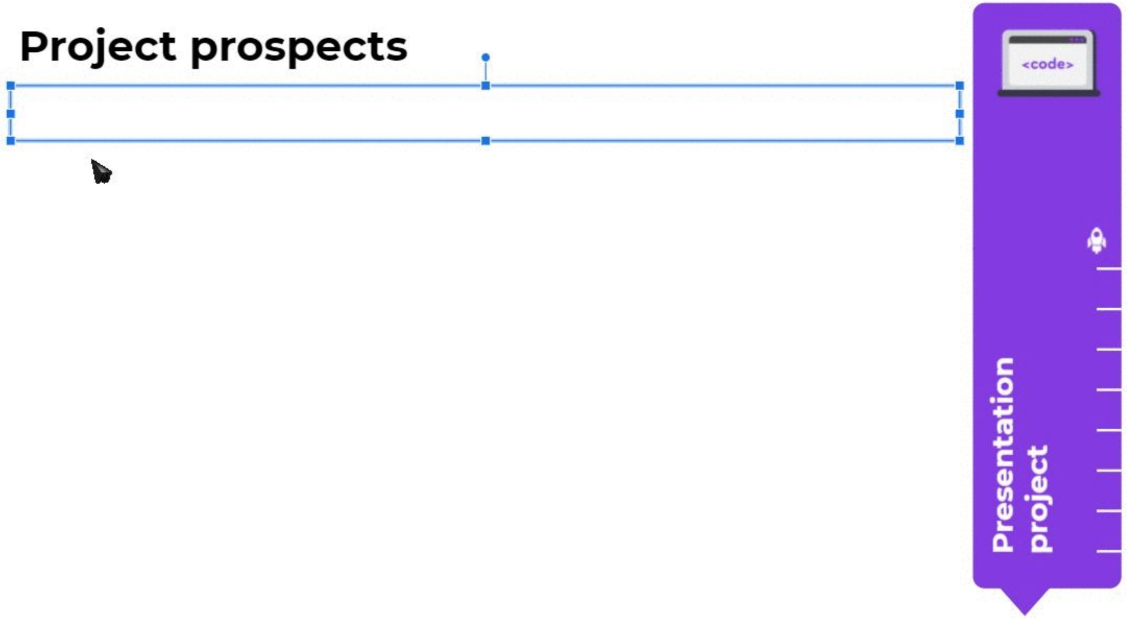
You can also copy and paste an image directly from the browser.





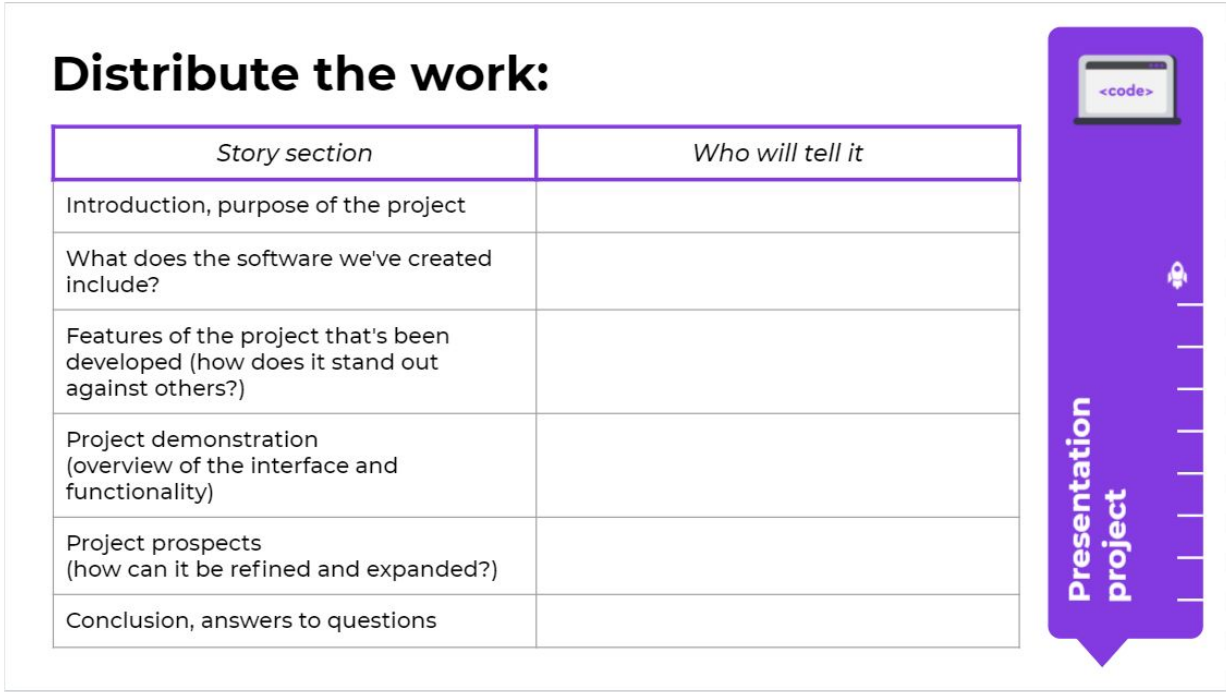
Google Slides: Getting started

Example of text editing and image placement.
A picture copied directly from the browser.



Distributing roles

Open the first (hidden) slide of the template. Distribute the slides.



Distribute the work:

Story section	*Who will tell it*
Introduction, purpose of the project	
What does the software we've created include?	
Features of the project that's been developed (how does it stand out against others?)	
Project demonstration (overview of the interface and functionality)	
Project prospects (how can it be refined and expanded?)	
Conclusion, answers to questions	

Brain storm