algorithmics

Module 5. Lesson 2.

Maze Game. Part 1

Link to the methodological guidelines

Discussion:

Working on the project

Developers, a new order ’s come in !

Our department is developing the **Maze game**.

The game designers have already thought out the logic of the game, and the artists have drawn the images for the background and sprites.

Now, the developers have to program the game functionality.

Let’s study the technical specification in more detail!

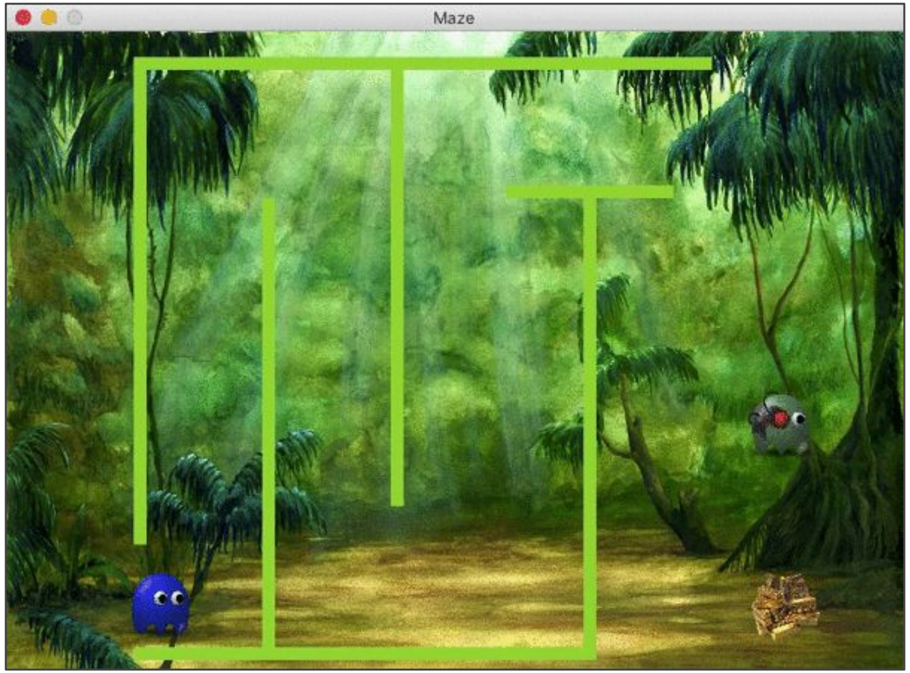
Cole,
Senior Developer

Discussion of work tasks

Technical specifications

The **goal** is to program the functionality of the Maze game.

Expected game appearance:



Discussion of work tasks

Technical specifications

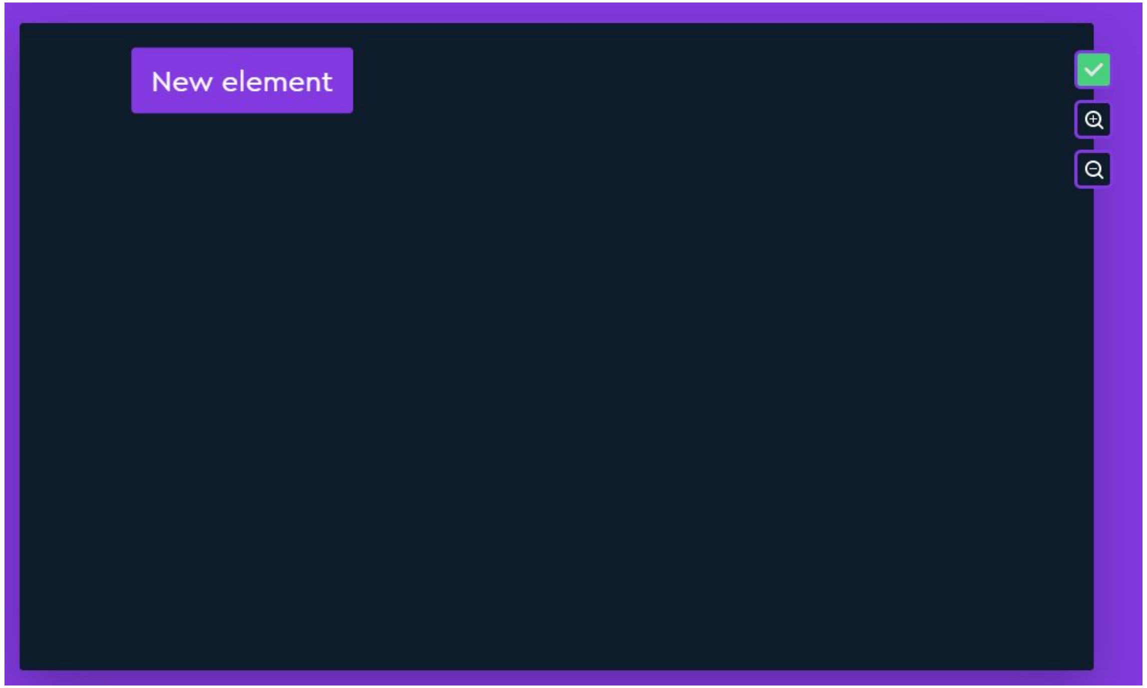
The **goal** is to program the functionality of the Maze game.

Requirements:

- ❏ A design that matches the one in the picture.
- ❏ The ability to use keys to control the player sprite.
- ❏ Presence of obstacles for the player:
  - ➔ maze walls (at least three);
  - ➔ an enemy sprite guarding the treasure.
- ❏ The player wins if they go through the maze without touching the walls, get past the enemy, and touch the treasure.
- ❏ The player loses immediately if they either touch the walls of the maze or collide with the enemy.

Planning t he work: mind map

Let's build the project's mind map:

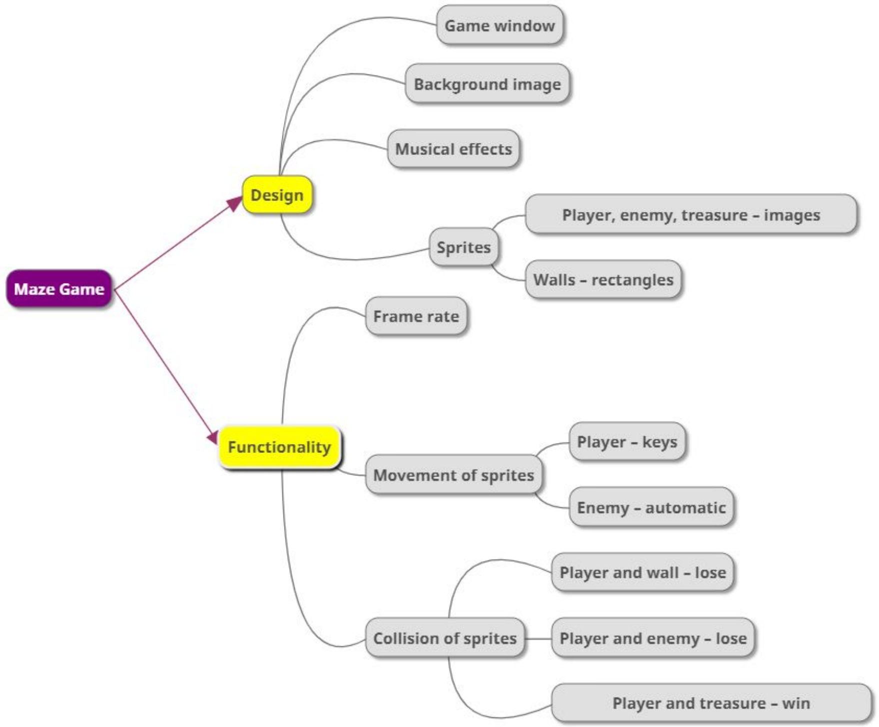


Open the exercise with the map on the platform. What will be the content of the game?

Discussion of work tasks

Planning the work: mind map

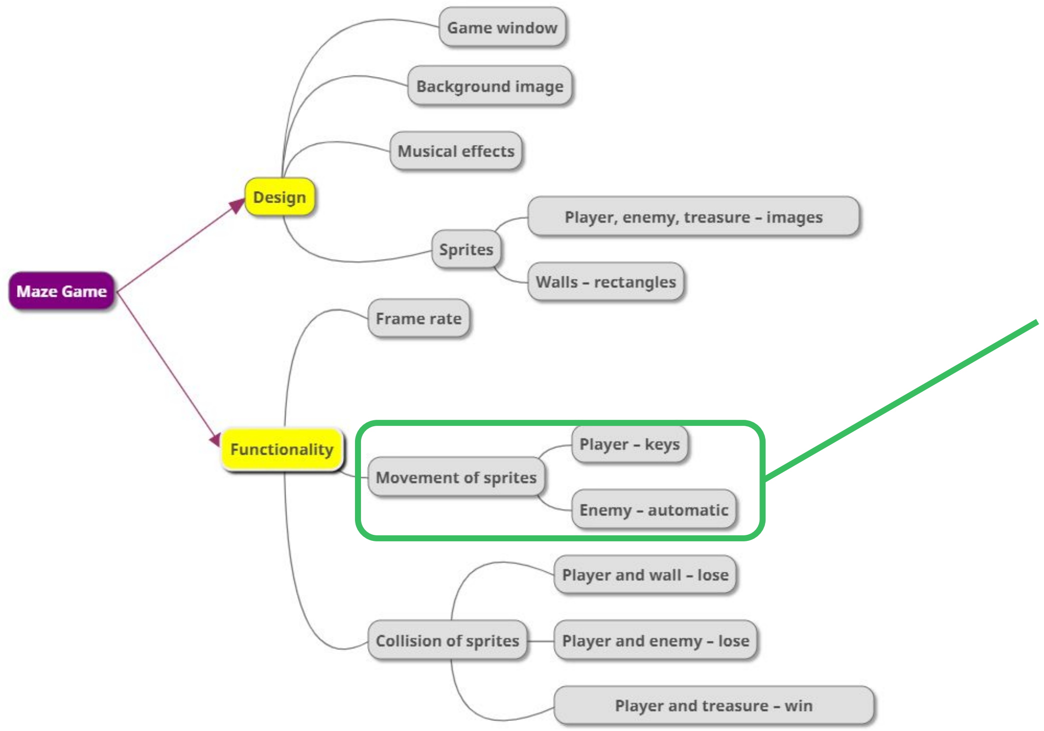
The mind map of the project by developer Cole:



Discussion of work tasks

Planning the work: mind map

The mind map of the project by developer Cole:



How do we create objects of the same kind (character) but with different methods?

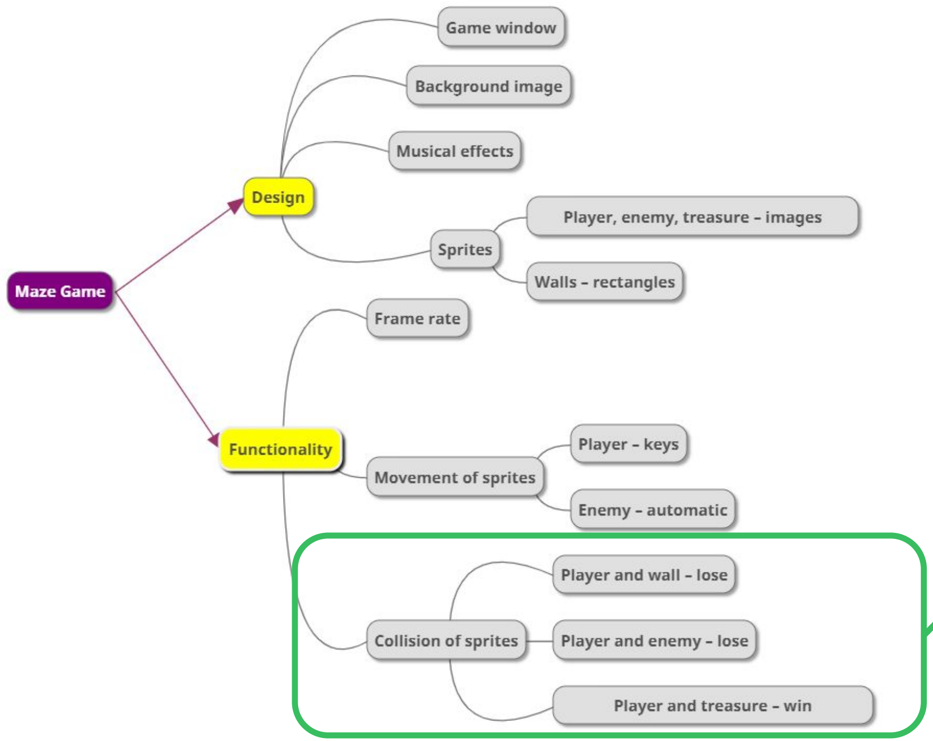
How do we give each sprite its own way of moving?

What if there are several enemies?

Discussion of work tasks

Planning the work: mind map

The mind map of the project by developer Cole:



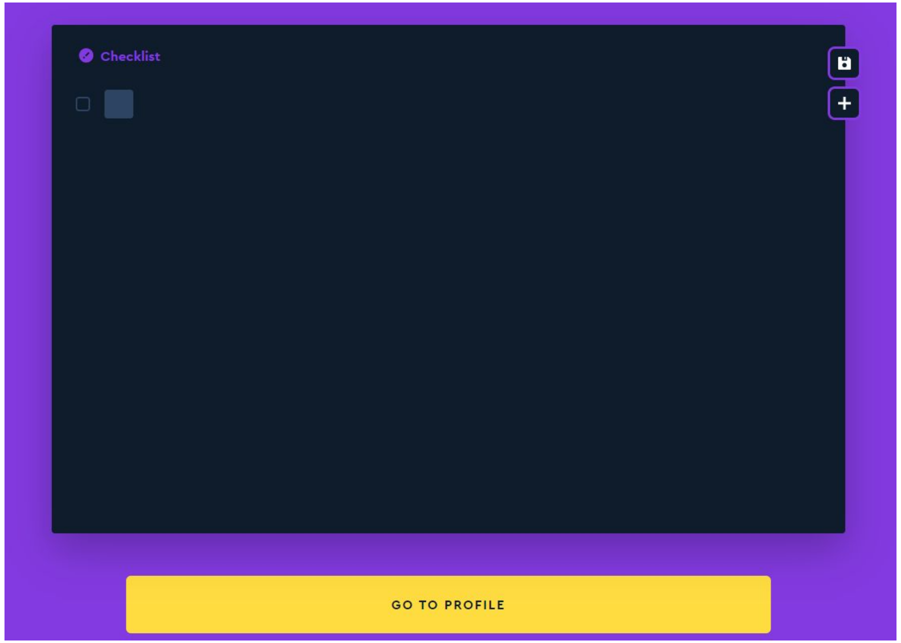
We haven't programmed **sprite collision** yet...

Is there a ready-made tool in PyGame for that?

Discussion of work tasks

Planning the work: checklist

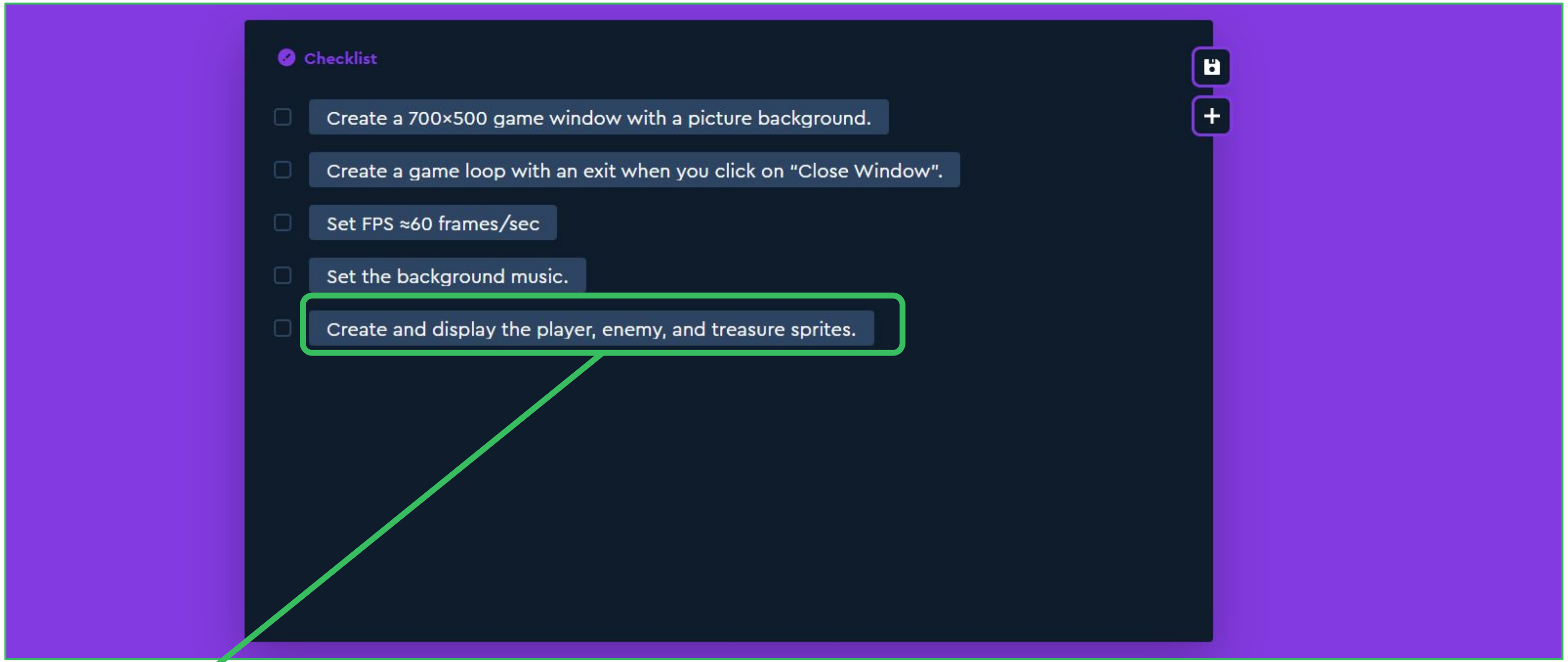
We will fill out the checklist as we mark off the project tasks:



What tasks are we implementing today?

Planning the work: checklist

Checklist of tasks for today, as suggested by Cole:



Checklist

- [ ] Create a 700×500 game window with a picture background.
- [ ] Create a game loop with an exit when you click on "Close Window".
- [ ] Set FPS ≈60 frames/sec
- [ ] Set the background music.
- [ ] Create and display the player, enemy, and treasure sprites.

It may be worth considering how we'll use these sprites in the future.
Setting a subscription for all possible collisions is not easy...

The goal of the workday is

to program the Maze game window and place the characters in it.

Today, you will:

- review how to create games using PyGame.
- review object-oriented programming and use it to create sprites.
- program the game template with backgrounds, characters, and music.

Discussion of work tasks

Qualifications

Show your knowledge

**of the basics of game development
and object-oriented programming**

Qualifications

How do we create a game window ?

How do we set a <u>background for it with a picture</u> ?

What if the picture doesn’t fit in the window?

Qualifications

Creating the background template

Note. The picture must be in the project folder.

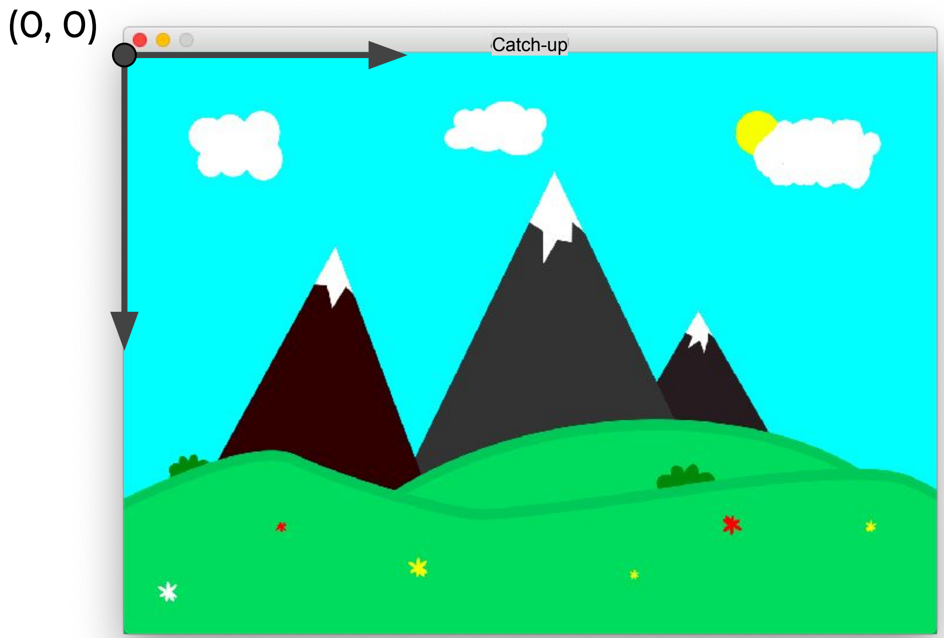
Command	*Purpose*
`window = display.set_mode((700, 500))`	Create a window with size: (width, length).
`display.set_caption("Catch-up")`	Set the window title.
`background = transform.scale( image.load("background.png"), (700, 500) )`	Create a picture object; adapt the picture size to the window parameters.
`window.blit(background,(0, 0))`	Display the background picture in the window.

Qualifications

Creating the background template

In PyGame, the origin is located in the upper left corner of the window.

The size of the window is determined by the developer.

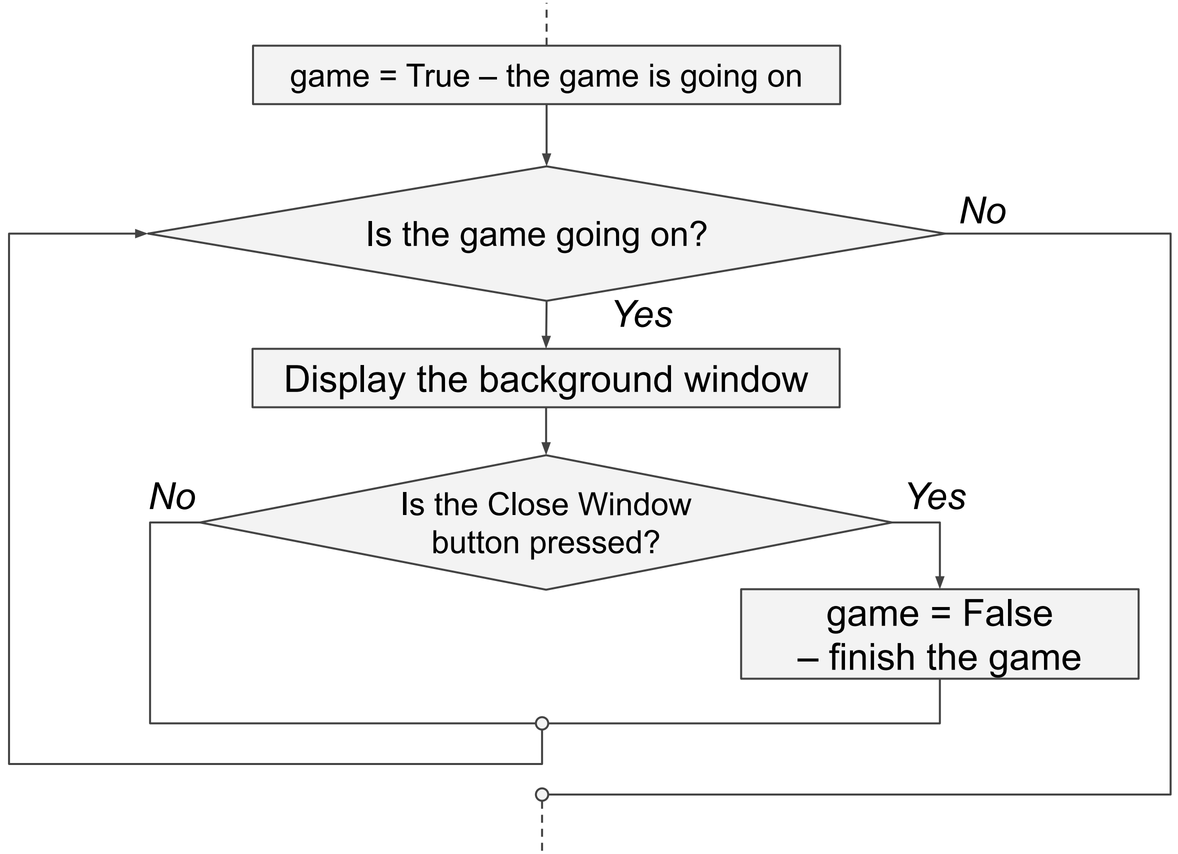


What is a game loop ?

How to <u>create</u> it?

What is <u>the condition for its ending up?</u>

Flowchart of the simplest game loop:


game = True – the game is going on
Is the game going on?
No
Yes
Display the background window
Is the Close Window button pressed?
No
Yes
game = False – finish the game

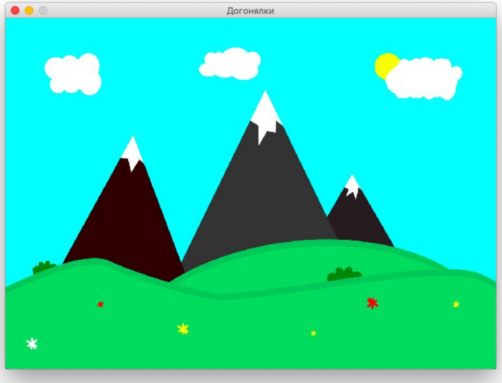
The program code that opens the game window and displays the picture:

```
from pygame import *
window = display.set_mode((700, 500))
display.set_caption("Catch-up")
background = transform.scale(image.load("background.png"), (700, 500))

game = True
while game:
   window.blit(background,(0, 0))

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

   display.update()
```



Frames that appear on the screen at every step of the loop must be updated.

What is a **class**?

What is **a class field/property** ?

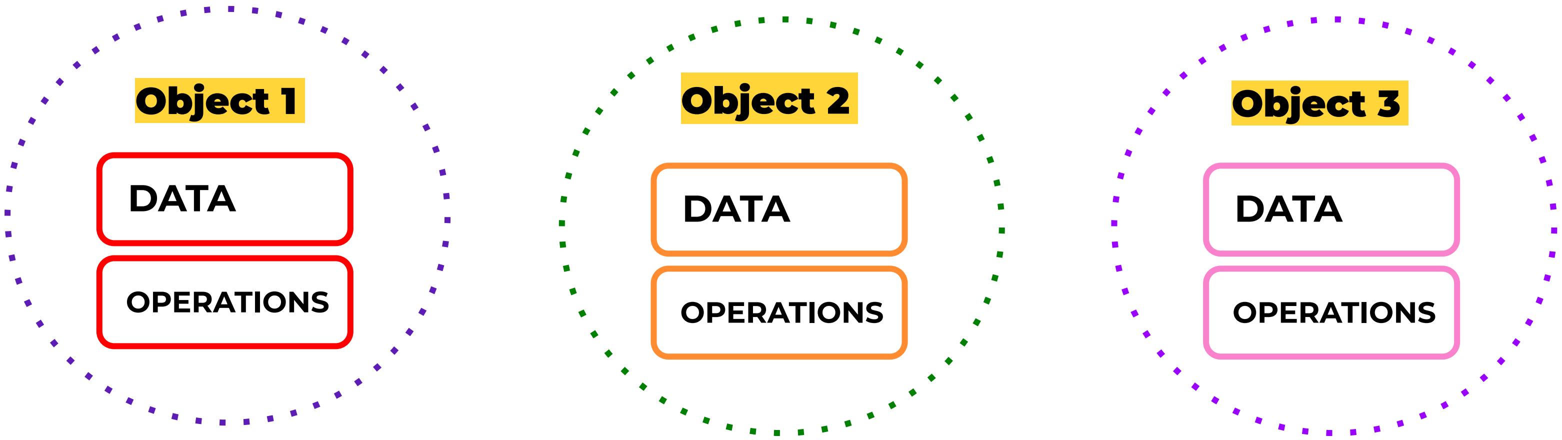
What is **a class method** ?

Property/field —

a variable placed inside an object.

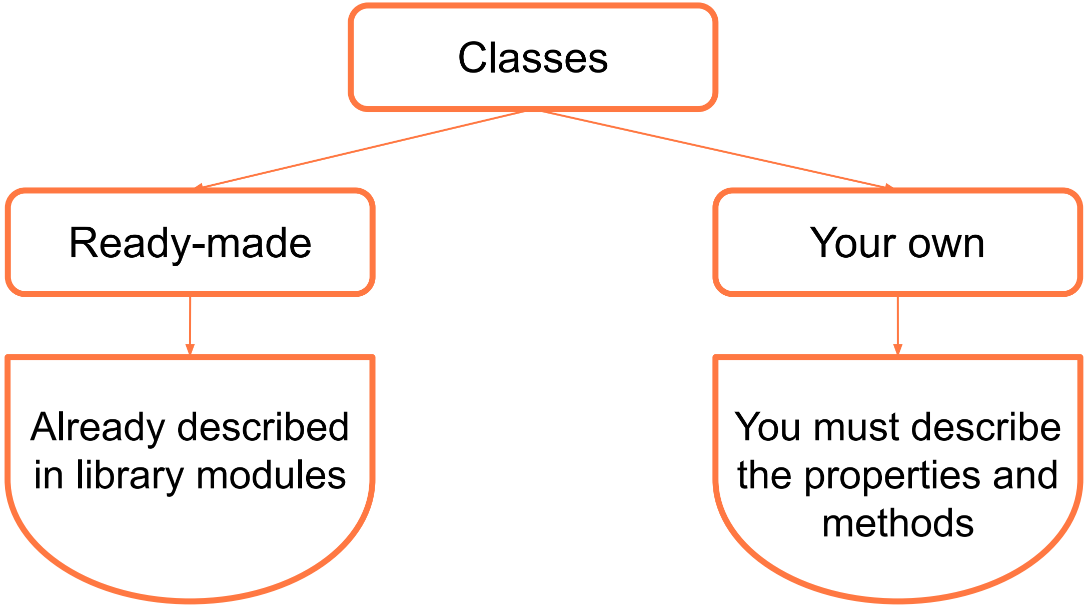
Method —

a function placed inside an object.



Class —

- ➢ a common name for many objects;
- ➢ in programming: a common description of how these objects should be arranged.


Classes
Ready-made
Your own
Already described in library modules
You must describe the properties and methods

How to create a class?

What is a class constructor ?

Qualifications

Creating classes

To create a class, we must:

- in the constructor, list the **properties** that determine the features for this instance of the class;
- list the **methods** for managing the instance.

```
class Class name():
    def __init__(self, Value):
        self.Property name = Value
    def Method name(self):
        Operation with the object and properties
        Operation with the object and properties
```

Creating classes

To create a class, we must:

- <u>in the constructor</u>, list the **properties** that determine the features for this instance of the class;
- list the **methods** for managing the instance.

```
class Class name():
    def __init__(self, Value):
        self.Property name = Value
    def Method name(self):
        Operation with the object and properties
        Operation with the object and properties
```

A special **constructor** function that creates an instance of a class with the specified properties.

```
__init__
```

Two underscores.

Qualifications

What is the meaning of inheritance ?

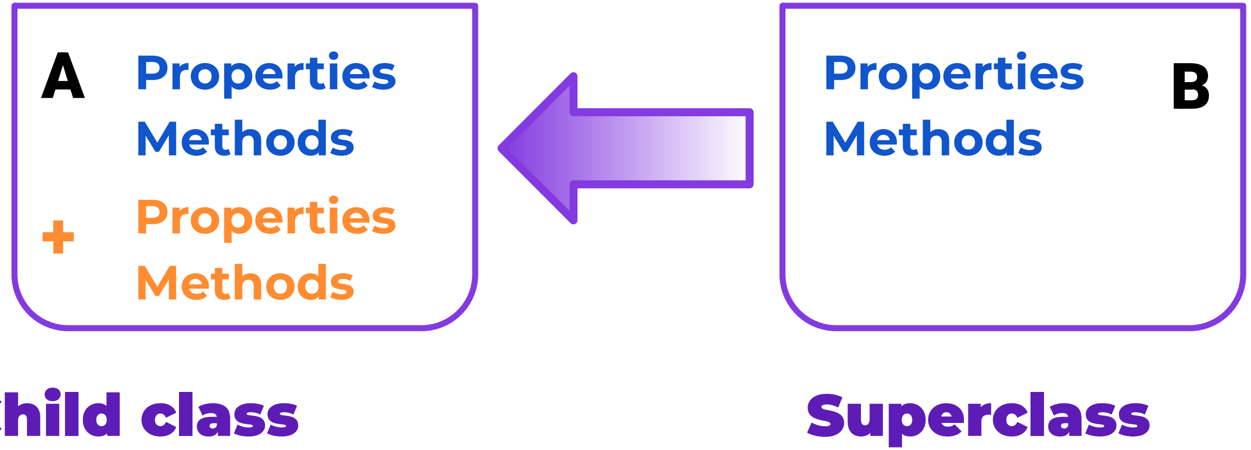
How to <u>create a child class</u> of an existing class?

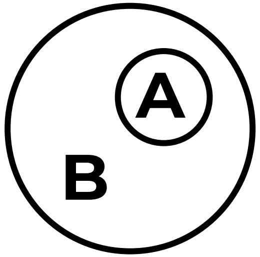
Give the real-life examples of <u>superclasses and child classes</u> .

Qualifications

Inheritance

Class inheritance helps to apply all the parameters specified before for a more general class to another, more specific class, which is called the child class.





Class A is nested within class B

Superclass and child class

Let's say the superclass is already written. In that case, to create a child class, we have to:

- when creating the child, specify the superclass name;
- supplement the child class with the necessary methods.

```
class Child class name ( Superclass name ):
    def __init__(self, Value ):
        super().__init__( Value )
    def Method name (self):
        Operation with the object and properties
```

An example in which ***no new properties are introduced****.*

The child class is only supplemented with a ***new method****.*

Qualifications

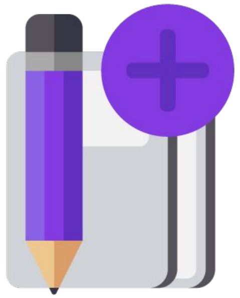
Qualifications confirmed!

Great! You are ready to brainstorming and complete your work task!

Module 5. Lesson 2. Maze Game. Part 1

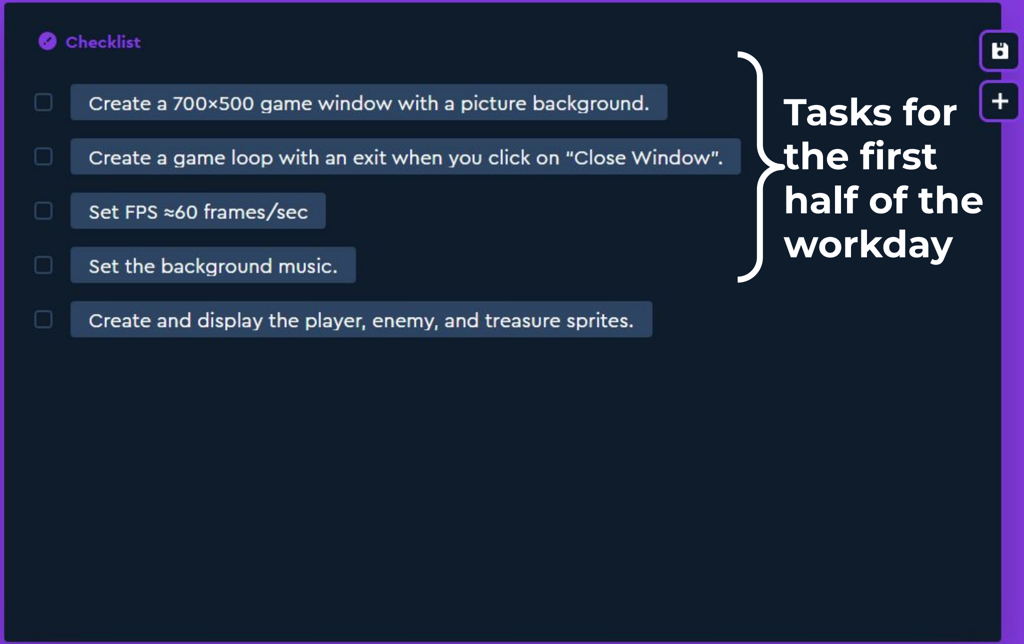
Brainstorm:

Maze Game Template



Let’s create the game template

Program the game scene with background music:



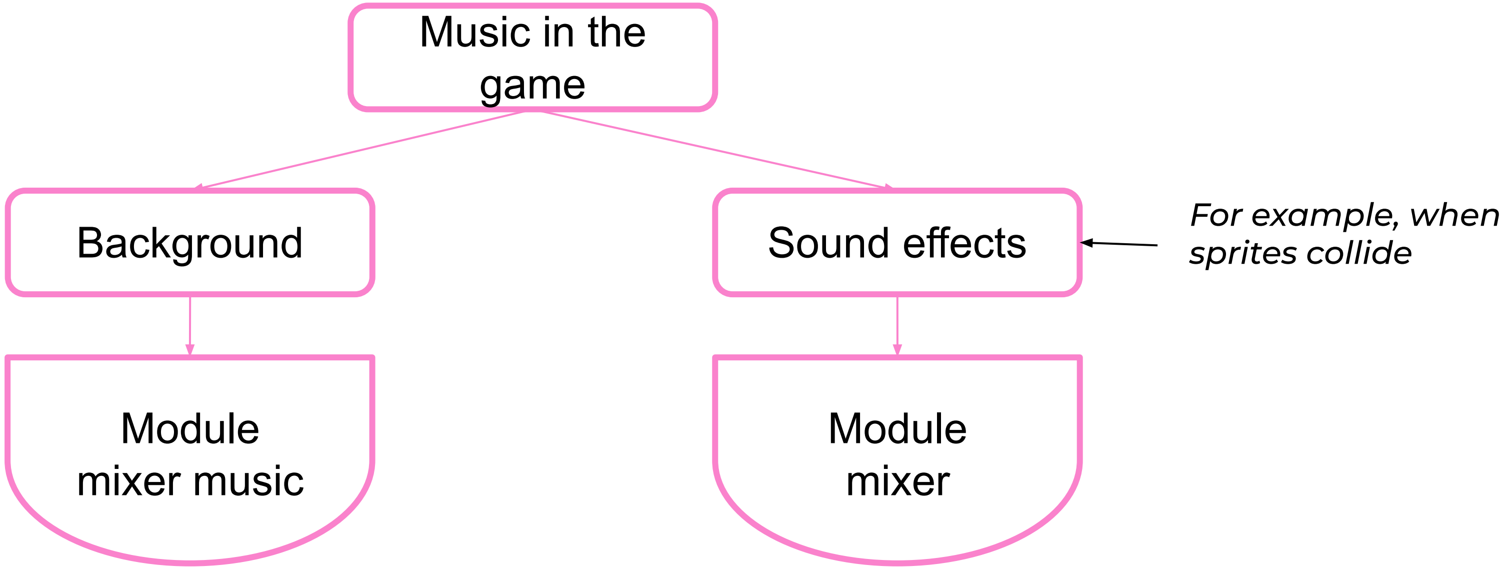
Checklist

- Create a 700×500 game window with a picture background.
- Create a game loop with an exit when you click on "Close Window".
- Set FPS ≈60 frames/sec
- Set the background music.
- Create and display the player, enemy, and treasure sprites.

Tasks for the first half of the workday

Musical soundtrack

The PyGame library has two modules for working with music:



All operating systems support the following formats: **ogg** and **wav**.

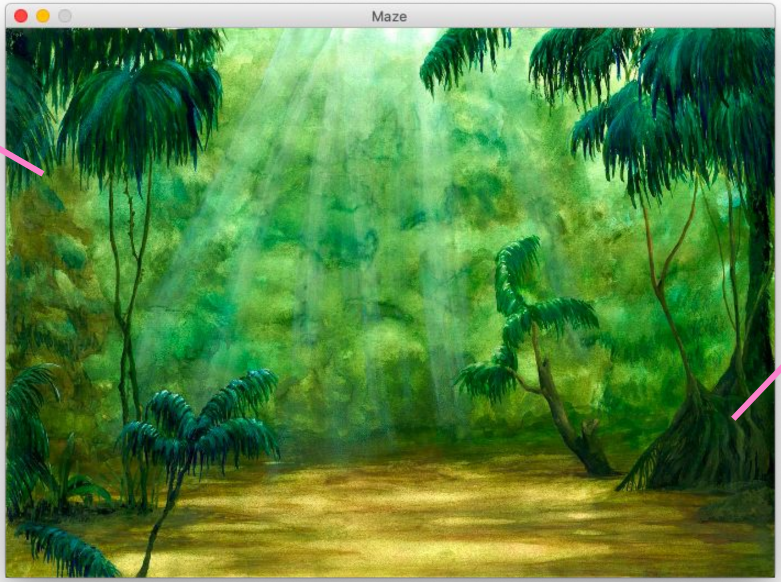
Methods for working with music

Command	*Purpose*
`mixer.init()`	Make it possible to use Mixer.
`mixer.music.load('jungles.ogg')`	Load a music file for background playback.
`mixer.music.play()`	Start playing background music.
`kick = mixer.Sound('kick.ogg')`	Load a sound for one-time playback (for example, when sprites collide).
`kick.play()`	Play back the sound.

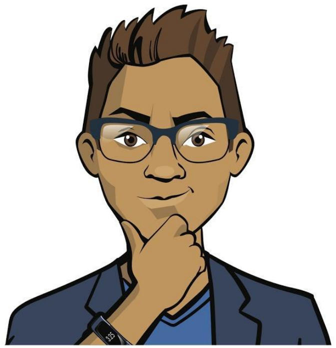
Brainstorming

How to program the creation of a game window with the picture and background music?

background.png



Background music: jungles.ogg



Brainstorming

Program outline:

The result – a game scene with background music.

Enabling PyGame modules

Creating objects for the background

Connecting and starting music

Variable game = True (game started)

Game loop:

End of the game if the Close Window button is pressed

Scene update
(next frame of the game loop)

Program outline:

The result – a game scene with background music.

Enabling PyGame modules

Creating objects for the background

Connecting and starting music

Variable game = True (game started)

Game loop:

End of the game, if the Close Window button is pressed

Scene update
(next frame of the game loop)

```
#Game scene:
win_width = 700
win_height = 500
window = display.set_mode(
    (win_width, win_height)
)
display.set_caption("Maze")
background = transform.scale(
    image.load("background.jpg"),
    (win_width, win_height)
    )
```

Program outline:

The result – a game scene with background music.

Enabling PyGame modules

Creating objects for the background

Connecting and starting music

Variable game = True (game started)

Game loop:

End of the game, if the Close Window button is pressed

Scene update
(next frame of the game loop)

```python
#music

mixer.init()

mixer.music.load('jungles.ogg')

mixer.music.play()
```

Program outline:

The result – a game scene with background music.

Enabling PyGame modules

Creating objects for the background

Connecting and starting music

Variable game = True (game started)

Game loop:

End of the game, if the Close Window button is pressed

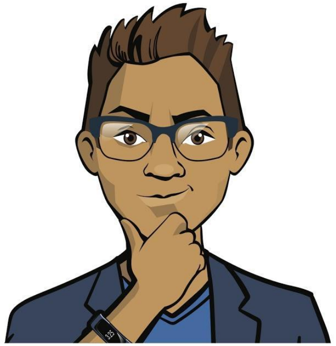
Scene update
(next frame of the game loop)

```
while game:
    for e in event.get():
        if e.type == QUIT:
            game = False
    window.blit(background,(0, 0))
    display.update()
    clock.tick(FPS)
```

Your tasks:

1. Program a game template with a picture background.
2. Set the background music.

If you have time left, choose the spots where future player and enemy sprites will appear on the scene.



Brainstorm

Module 5. Lesson 2. Maze Game. Part 1

Platform:

PyGame: Maze

Complete tasks in VS Code

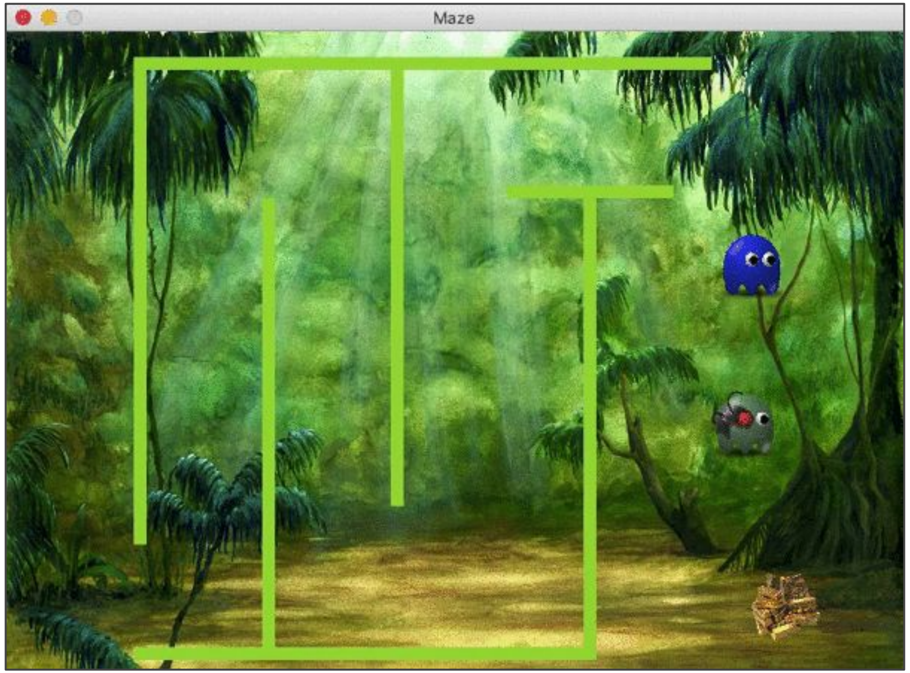
“PyGame: Maze”

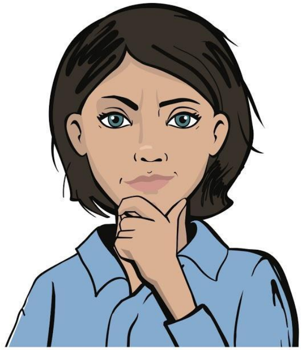
Brainstorming:

Classes for Sprites

Sprites collision

According to the technical specifications, the player loses *when they collide with an enemy sprite*. How can we program this?

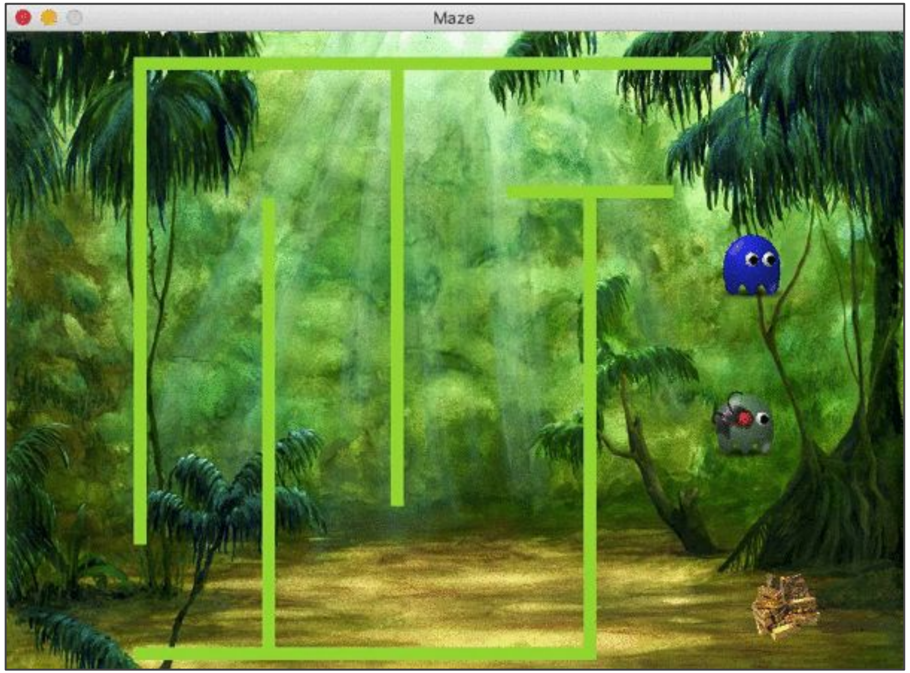


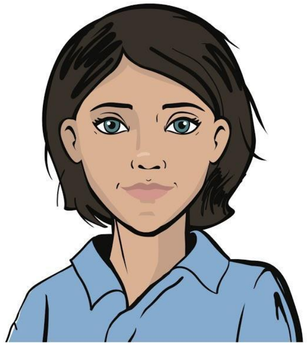


Brainstorming

Sprites collision

A possible solution: compare the coordinates of the current locations of the player and enemy sprites.

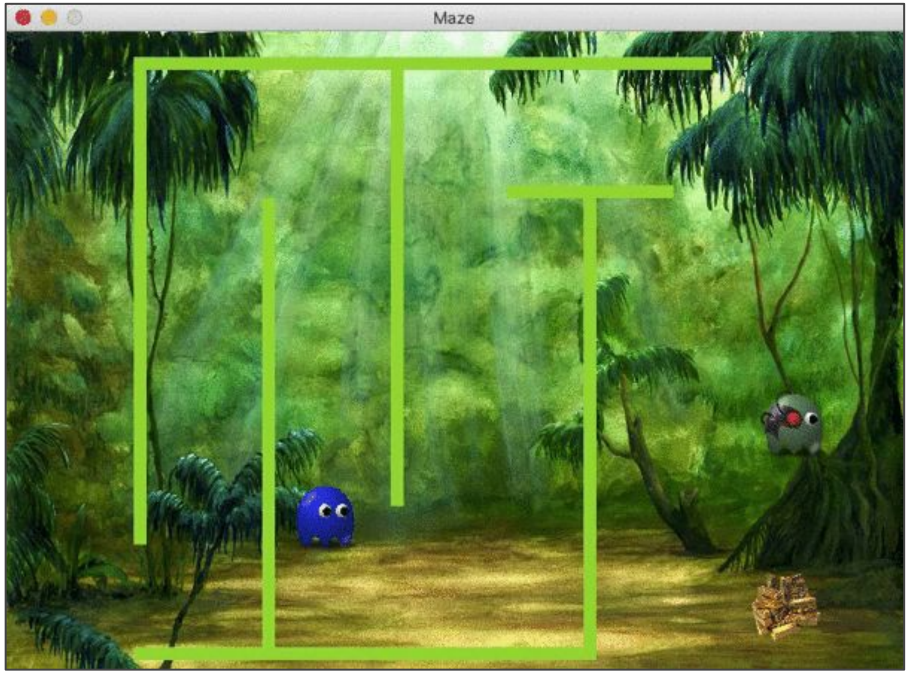


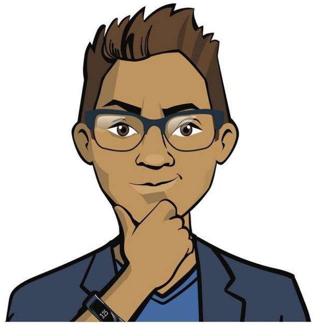


Brainstorming

Sprites collision

The player must also lose *when they collide with walls*. **Do we need** to compare the coordinates of the player with each wall?

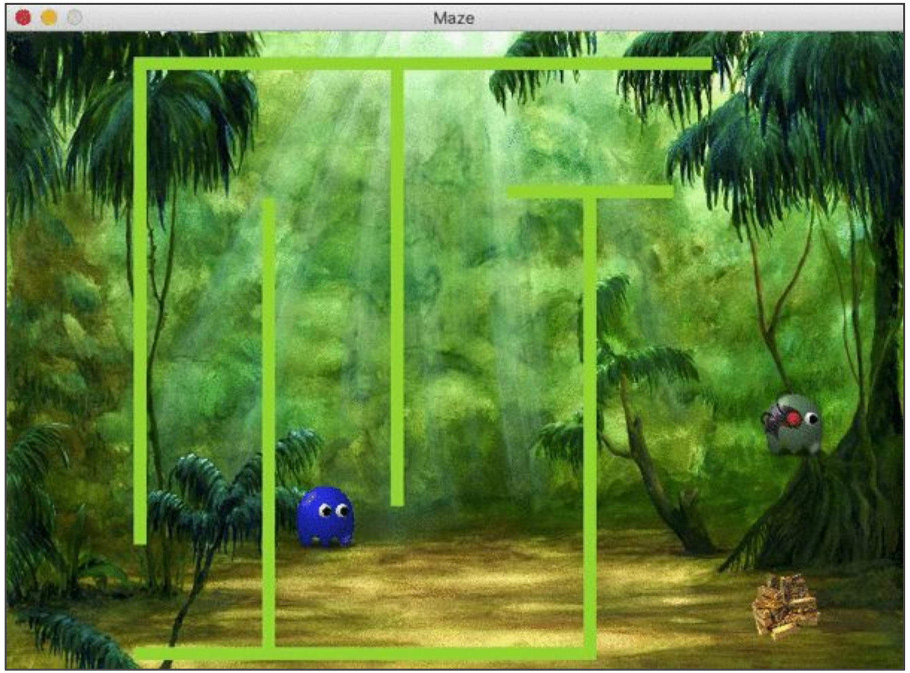


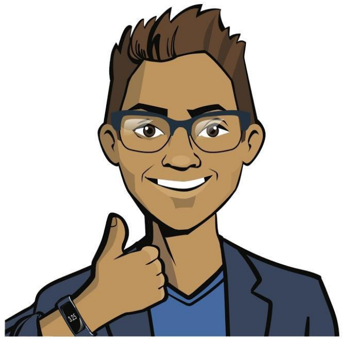


Brainstorming

Sprites collision

No! PyGame has a ready-made solution in the form of a ready-made Sprite class method!





Brainstorming

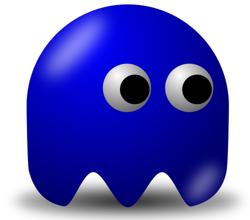
Creating sprites using classes

The base for creating sprites is already implemented in **the Sprite class**.

But some methods are universal for all sprites, while others are implemented differently depending on the type of sprite.

universal

individual for every sprite



sprite

Fits in some rectangle → What size does it have?

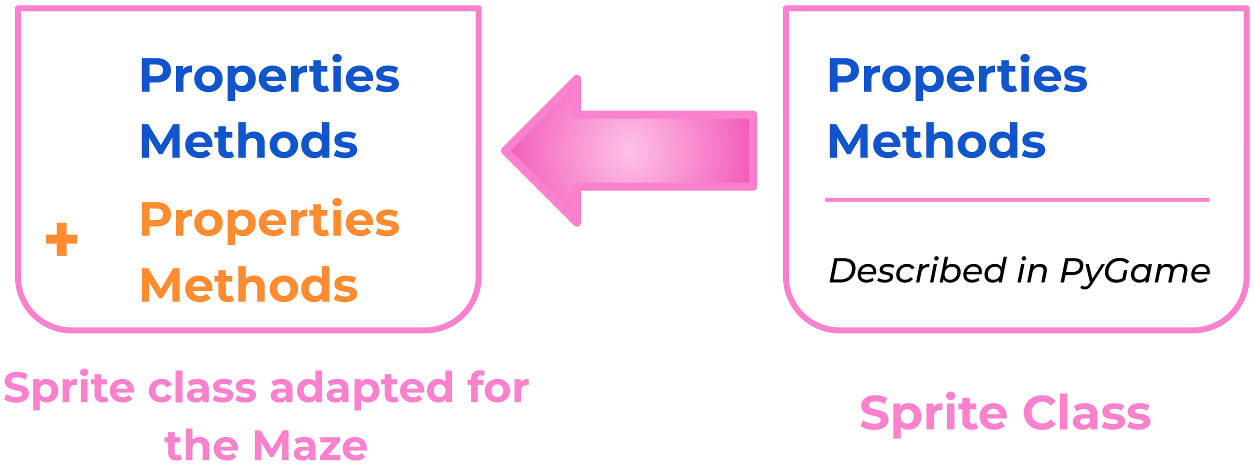
Can move with frame changes → How exactly does its position change?

Detects collisions with other sprites

Brainstorming

Creating sprites using classes

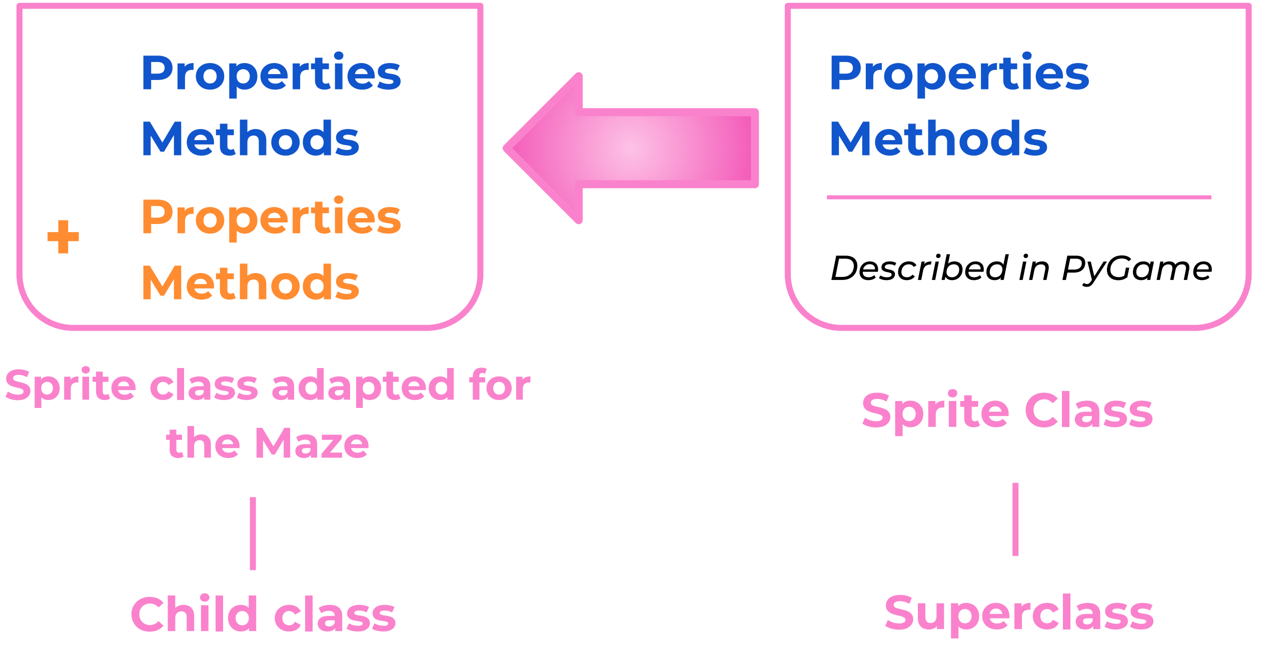
Is there a way to take useful properties and methods from the ready-made Sprite class and supplement them with new details for our game?



Creating sprites using classes

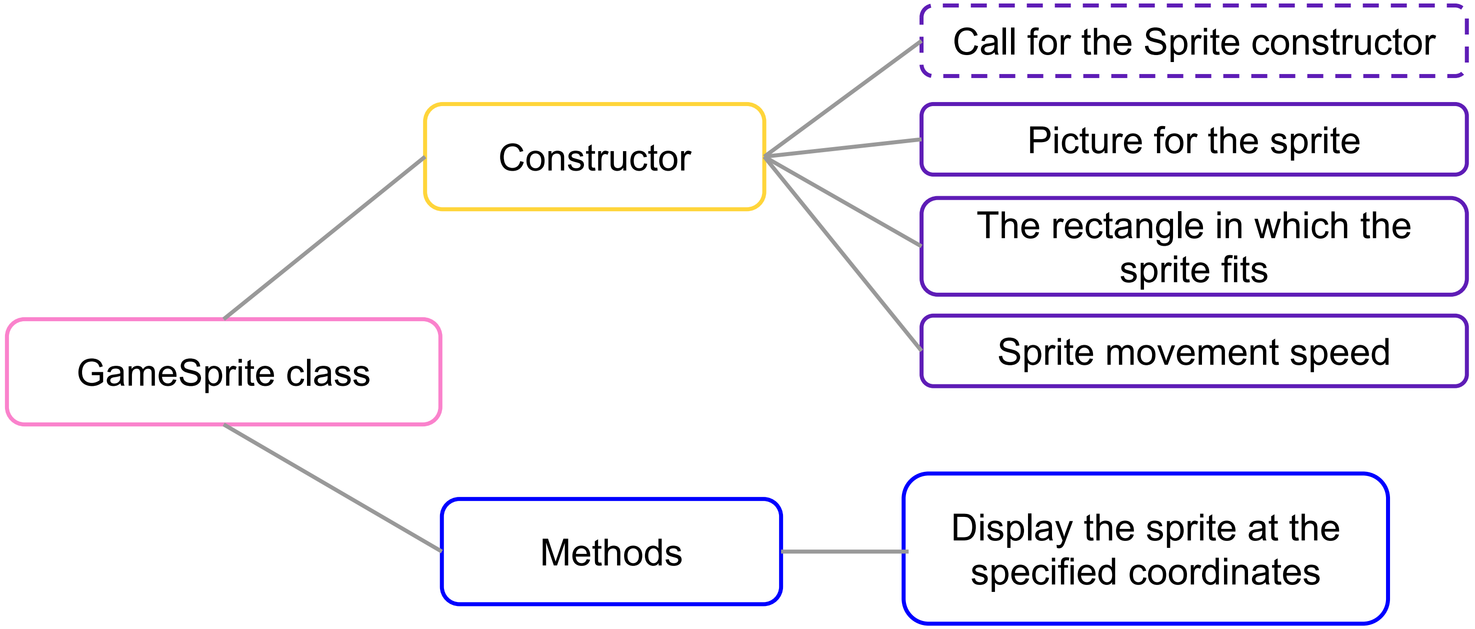
Yes! We’ll create the **GameSprite** child class of the ready-made **Sprite** class.

In GameSprite, we’ll specify the properties and methods to fit the particular features of Maze.



GameSprite class

GameSprite inherits from Sprite:


GameSprite class
Constructor
Call for the Sprite constructor
Picture for the sprite
The rectangle in which the sprite fits
Sprite movement speed
Methods
Display the sprite at the specified coordinates

The movement of sprites is a tricky task that we will discuss later.

Brainstorming

GameSprite class

```
class GameSprite(     ?     ):
    def __init__(self,     ?     ):
        super().__init__()
        self.image =     ?
        self.speed =     ?
        self.rect = self.image.get_rect()
        self.rect.x =     ?
        self.rect.y =     ?
    def reset(self):
            ?
```

Sprite position on the scene

Sprite class from the sprite module

Borrowing properties and methods from the superclass

Determining property values (from where?)

How do we position the sprite using coordinates?

Brainstorming

GameSprite class

```python
class GameSprite(sprite.Sprite):
    def __init__(self, player_image, player_x, player_y, player_speed):
        super().__init__()
        self.image = transform.scale(image.load(player_image), (65, 65))
        self.speed = player_speed
        self.rect = self.image.get_rect()
        self.rect.x = player_x
        self.rect.y = player_y
    def reset(self):
        window.blit(self.image, (self.rect.x, self.rect.y))
```

How do we create instances for the player, enemy, and treasure, and position them on the scene?

Program scheme:

The result – game scene with background music and positioned sprites

GameSprite class description

Creating objects for the background

Connecting and starting music

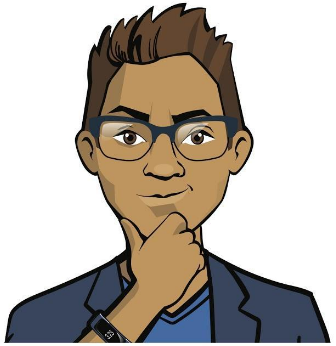
Creating sprites: *player*, *enemy* and *treasure*

Game loop:

- End of the game, if the Close Window button is pressed
- Rendering the scene and sprites on it
- Scene update (next frame of the game loop)

Your tasks is to:

1. Implement the GameSprite class.
2. Create instances of the GameSprite class for the sprites: player, enemy, and treasure.
3. Position the sprites on the scene.



Module 5. Lesson 2. Maze Game. Part 1

Platform:

PyGame: Maze

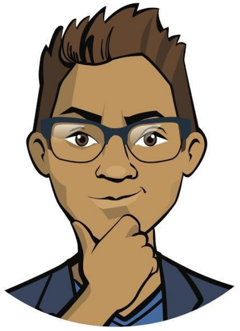
Complete tasks in VS Code

“PyGame: Maze”

Wrapping up the workday

To finish the workday, complete a technical interview:

1. What PyGame class did you learn about today? How will it help you create your game?
2. How do we create an child class of an existing class?
3. How do we set background music in the game?


Cole,
Senior Developer

Emily,
Project Manager

Great work!

Dear colleagues!

You’ve done an excellent job today.

In the next workday, we will continue working on the Maze game and determine the movement of our sprites!