algorithmics

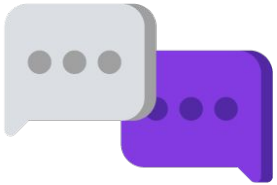
Module 5. Lesson 3.

The Maze game. Part 2

Link to guidelines 

Discussion:

Project work



Let’s continue working on the game

Our department has been creating the **Maze game**.

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

We have to program the functionality of the game.

Cole,
senior developer

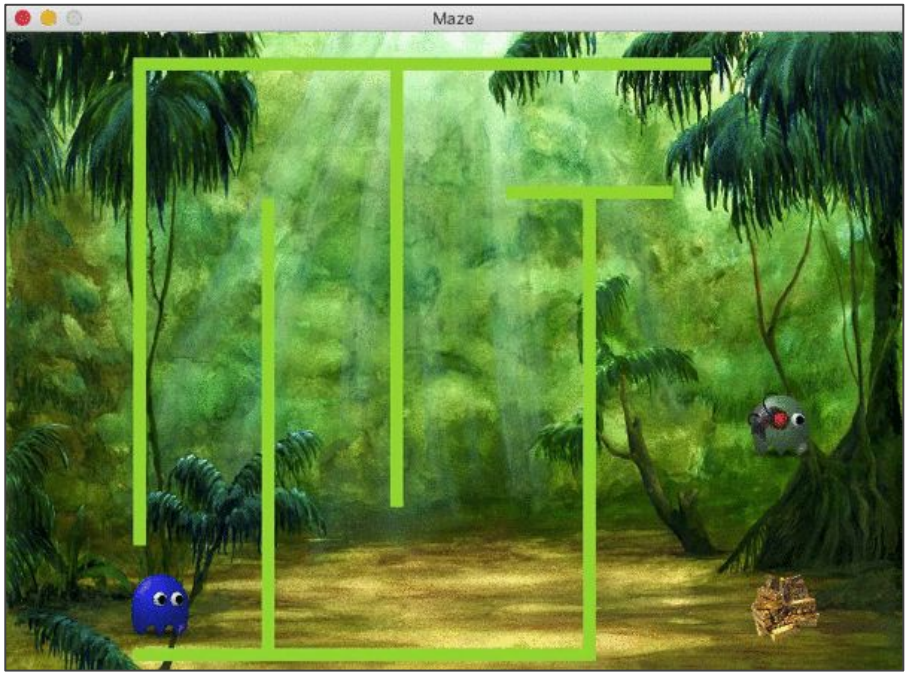
Let's recall our project tasks using a mind map and add the tasks for today to our checklist.

Discussing work tasks

Technical task

Goal — to program the functionality of the Maze game.

The expected look of the game:



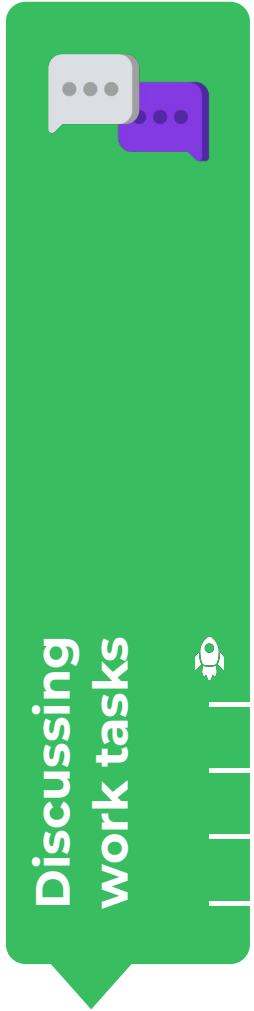
Discussing work tasks

Technical task

Goal — to program the functionality of the Maze game.

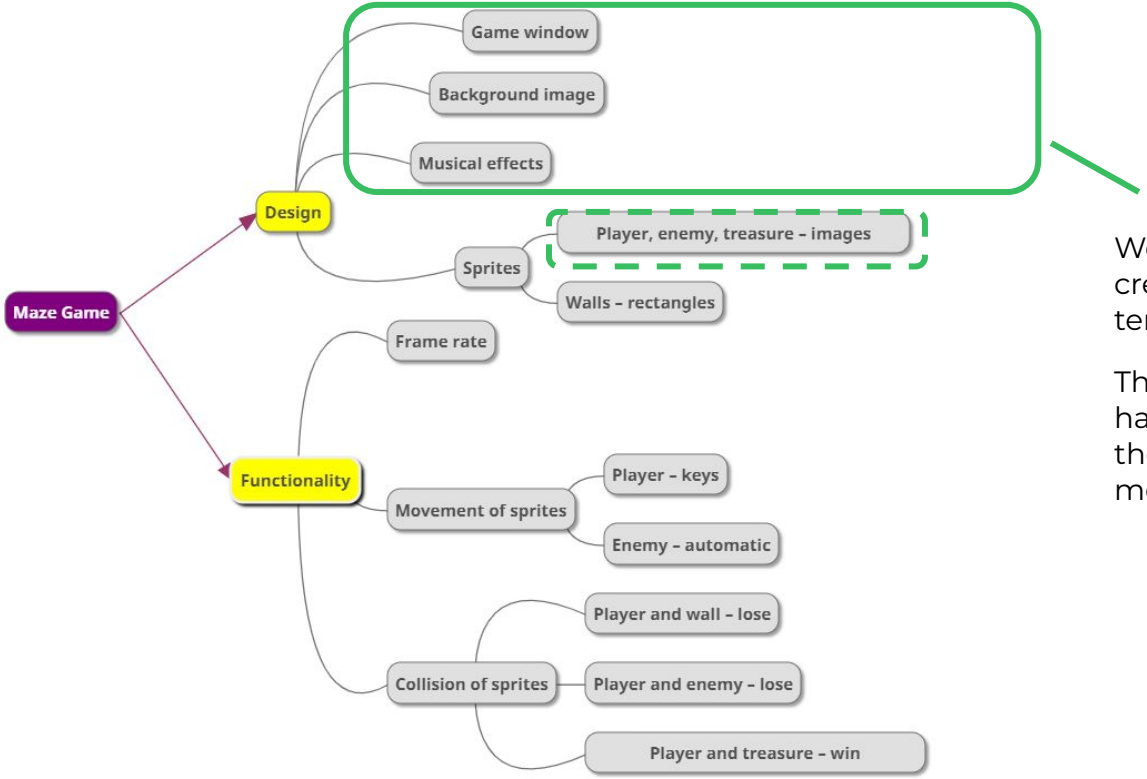
Requirements:

- ❑ The design should be like in the picture.
- ❑ Key control for the sprite player.
- ❑ Obstacles for the player:
  - ➔ mazewalls (at least three);
  - ➔ sprite enemy guarding the treasure.
- ❑ The player wins if they get through the mazewithout hitting the walls, bypass the enemy, and touch the treasure.
- ❑ The player immediately loses if they either touch the walls of the mazeor collide with the enemy.



Planning work : mind map

Mind map for the project from the developer Cole:



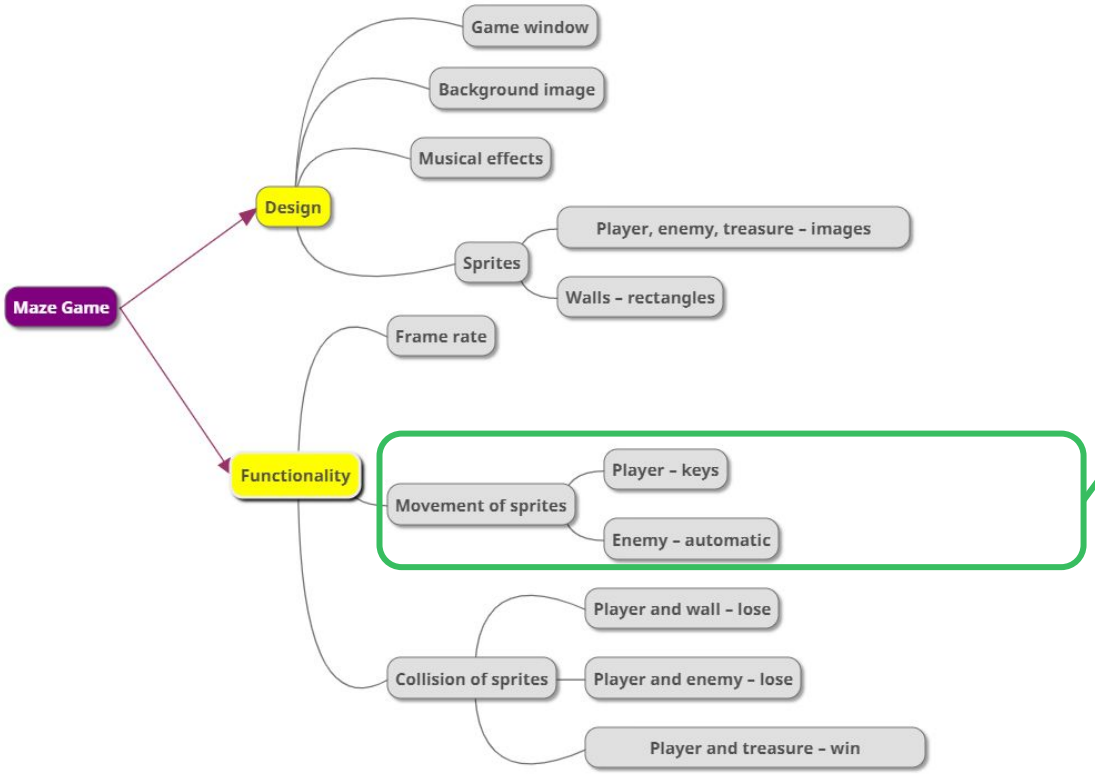
We have already created a game template.

The image sprites have been placed on the stage but not yet moved.

Discussing work tasks

Planning work: mind map

Mind map for the project from the developer Cole:



Let's program the sprite movement.

The player is controlled by keys, and the **enemy moves automatically**.

Discussing work tasks

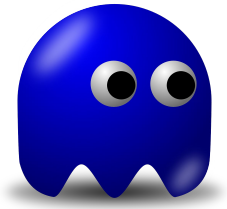
How do we set each type of sprite its own way to move around?

Both sprites are now instances of the GameSprite class.

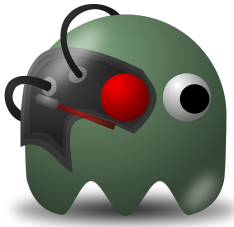
Properties
Methods

Already created

The GameSprite class



User controlled from the keyboard.

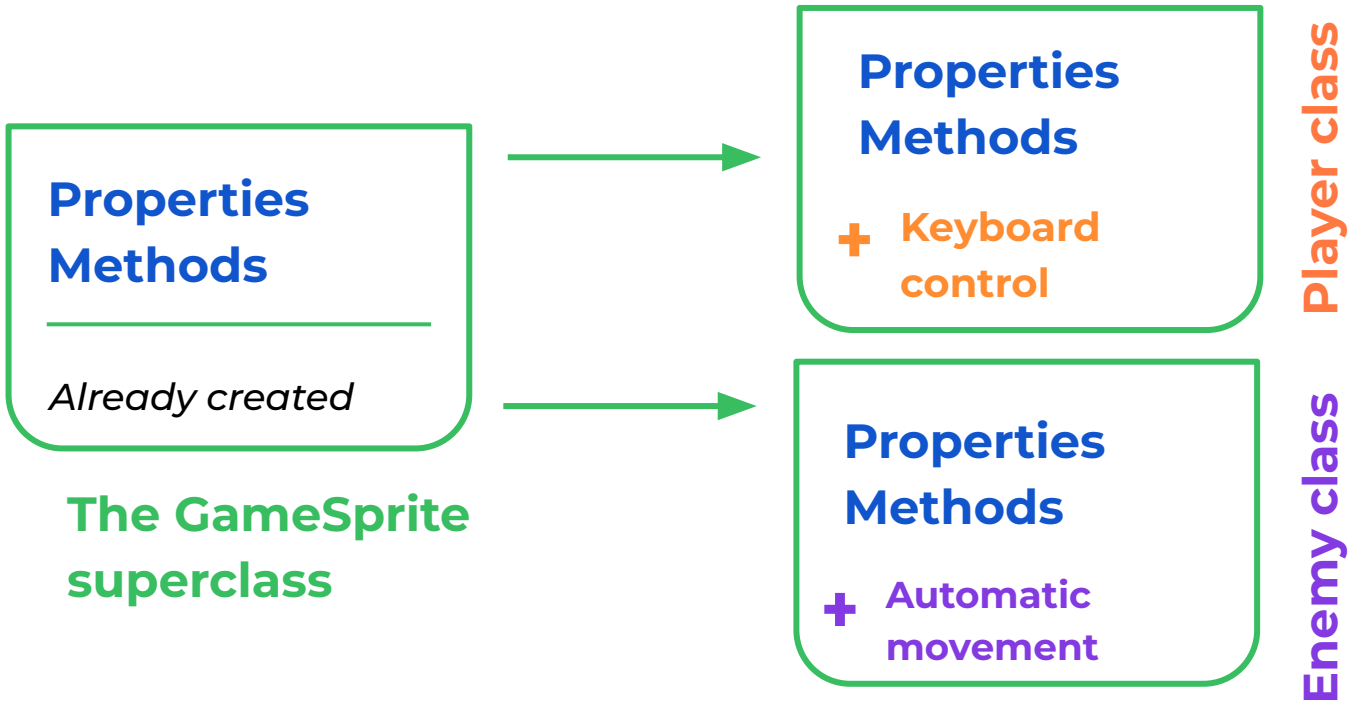


Moves left-right automatically.

Discussing work tasks

Moving different sprites

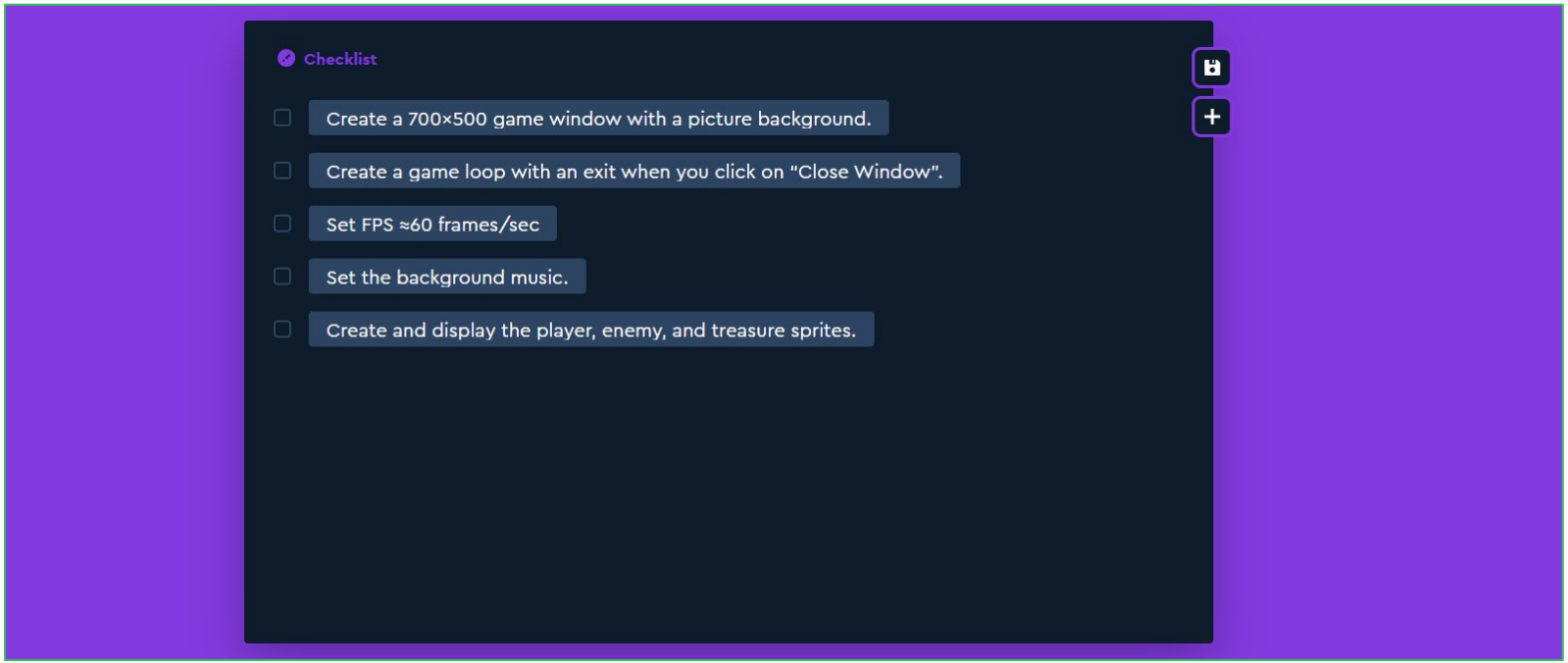
Let's create two child classes from GameSprite. They will inherit all the properties and methods of GameSprite, but they will move in their own way.



Discussing work tasks

Planning work: checklist

Let's add the tasks for today to the checklist.**Link to level**

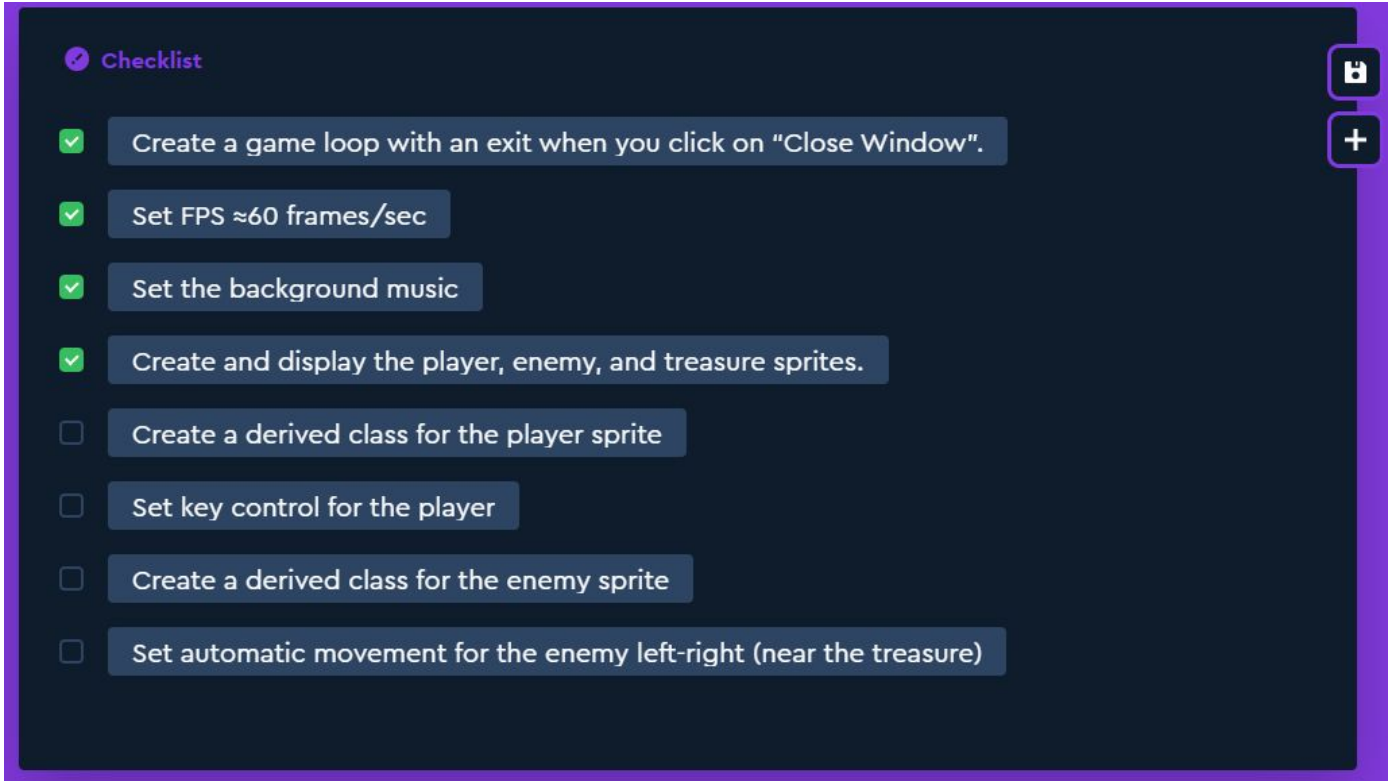


Don’t forget to check off completed tasks!

Discussing work tasks

Planning the work: checklist

Updated checklist from the developer Cole:



Discussing work tasks

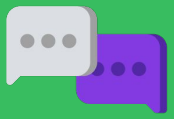
The goal of the work day is

to program classes for different types of sprites.

In the new classes, we will define movement methods for each type of character.

Today you will:

- Recall how to make games with PyGame.
- Recall inheritance and use it to revise the sprites.
- Program the key control for the player sprite and the automatic movement for the enemy sprite.



Discussing work tasks

Qualifications

Demonstrate your knowledge

of the basics of creating games and object-oriented programming

Qualifications

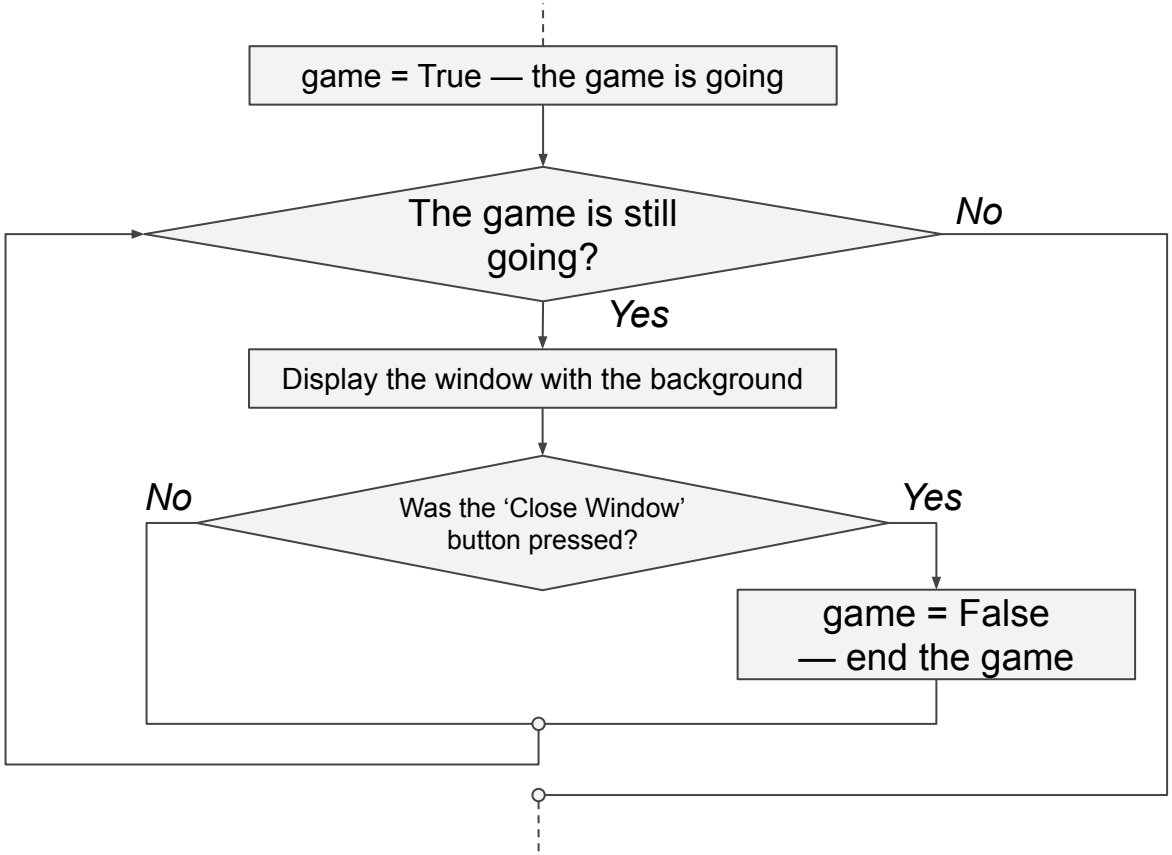
What is a game loop ?

How can we create it?

What is the condition for its ending up?

Qualifications

The work flow for the simplest game loop:



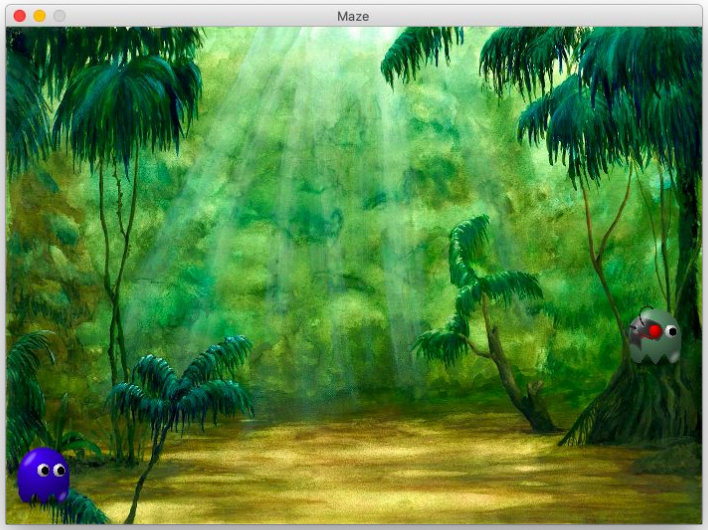
Qualifications

Code for the game loop of the Maze game:

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

    window.blit(background,(0, 0))
    player.reset()
    monster.reset()

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



The frames that appear on the screen need to be updated with each step of the loop.

Qualifications

What is the meaning of inheritance ?

How do we create a child class from an existing class?

Give the real-life examples of superclasses and child classes .

Qualifications

Inheritance

Class inheritance helps us transfer all the skills previously written for a more general class to another, more private class — a child class.

Sprite class adapted for the Maze game

Child class

Sprite class

Superclass

Qualifications

Superclass and child class

Let's suppose the superclass is already written. Then we will need to do the following to create a child class:

- when creating an inheritor, specify the name of the superclass;
- add the necessary methods to the child class.

```
class Inheritor name ( Superclass name ):
    def __init__(self, Value ):
        super().__init__( Value )
    def Method name (self):
        Action with object and properties
```

*Option in which **no new properties are introduced**.*

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

Qualifications

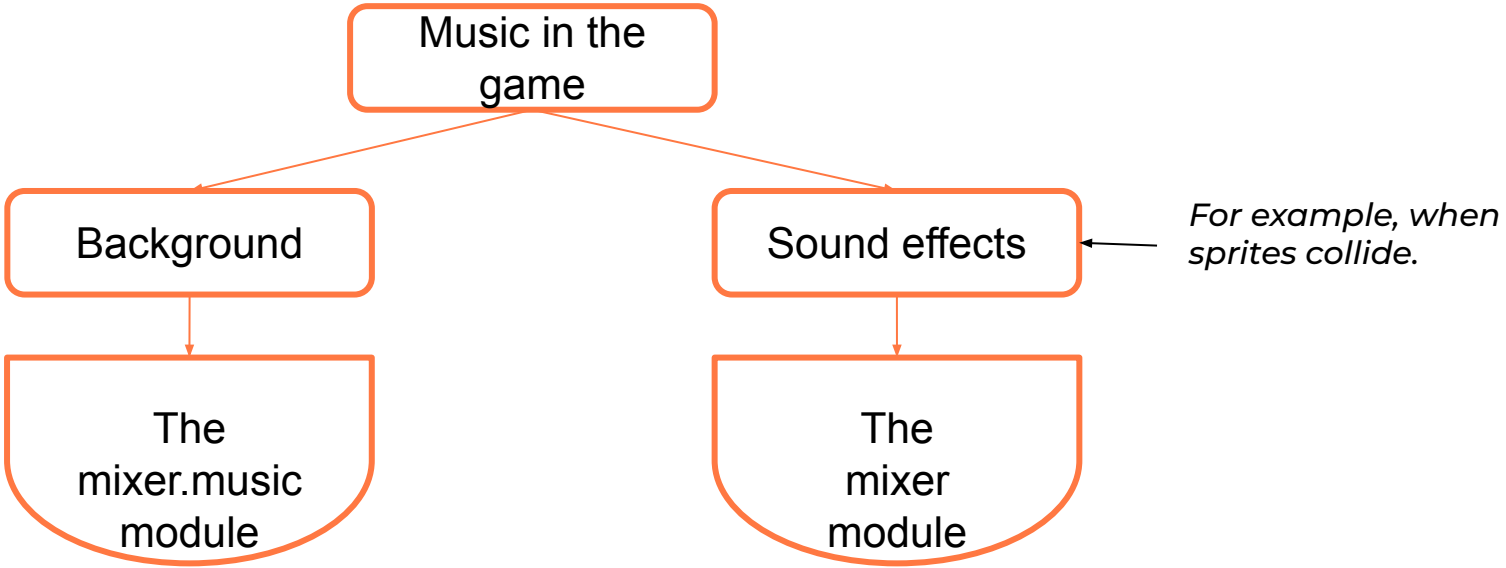
How do we set musical accompaniment ?

How do we set up background music ?

Qualifications

Musical accompaniment

There are two modules for working with music in the PyGame library:



For example, when sprites collide.

The following formats are supported on all operating systems: **ogg** and **wav**.

Methods for working with music

Command	*Purpose*
`mixer.init()`	Enable the use of Mixer.
`mixer.music.load('jungles.ogg')`	Load music files for background playback.
`mixer.music.play()`	Start playing background music.
`kick = mixer.Sound('kick.ogg')`	Load sounds for one-time playback (for example, when sprites collide).
`kick.play()`	Play sounds.

Qualifications

Qualifications confirmed!

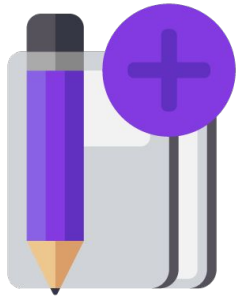
Great, you are ready to brainstorm and work on your tasks!

Qualifications

Module 5. Lesson 3. The Maze game. Part 2

Brainstorming:

The Player class



Let's program the Player class for the main player:

Checklist

- [x] Create a game loop with an exit when you click on "Close Window".
- [x] Set FPS ≈60 frames/sec
- [x] Set the background music
- [x] Create and display the player, enemy, and treasure sprites.
- [ ] Create a derived class for the player sprite
- [ ] Set key control for the player
- [ ] Create a derived class for the enemy sprite
- [ ] Set automatic movement for the enemy left-right (near the treasure)

Tasks for the first half of the work day

Brainstorming

Relationship between classes

The GameSprite class inherits from Sprite from PyGame.
The Player class inherits from GameSprite.

Properties
Methods

Universal sprites for any game

Sprite superclass (PyGame)

Properties
Methods

\+ **Properties**
Methods

Sprites that are universal for our game

GameSprite class (native)

Properties
Methods

\+ **Properties**
Methods

\+ **Methods**

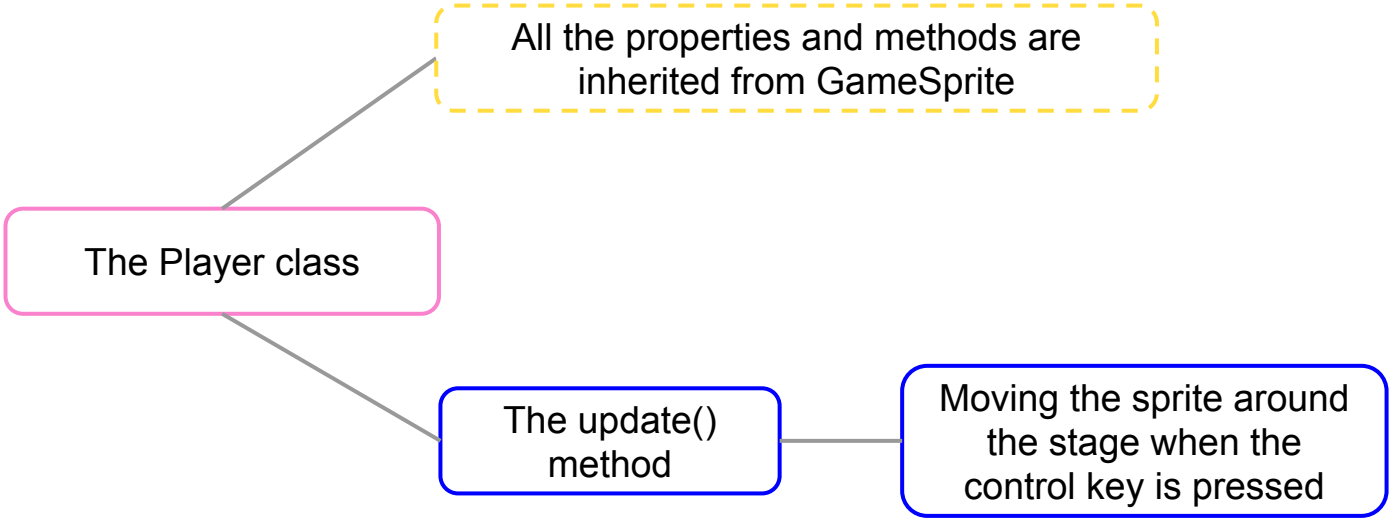
Specific type of sprite

Player class (native)

Brainstorming

The Player class

Player inherits from GameSprite:



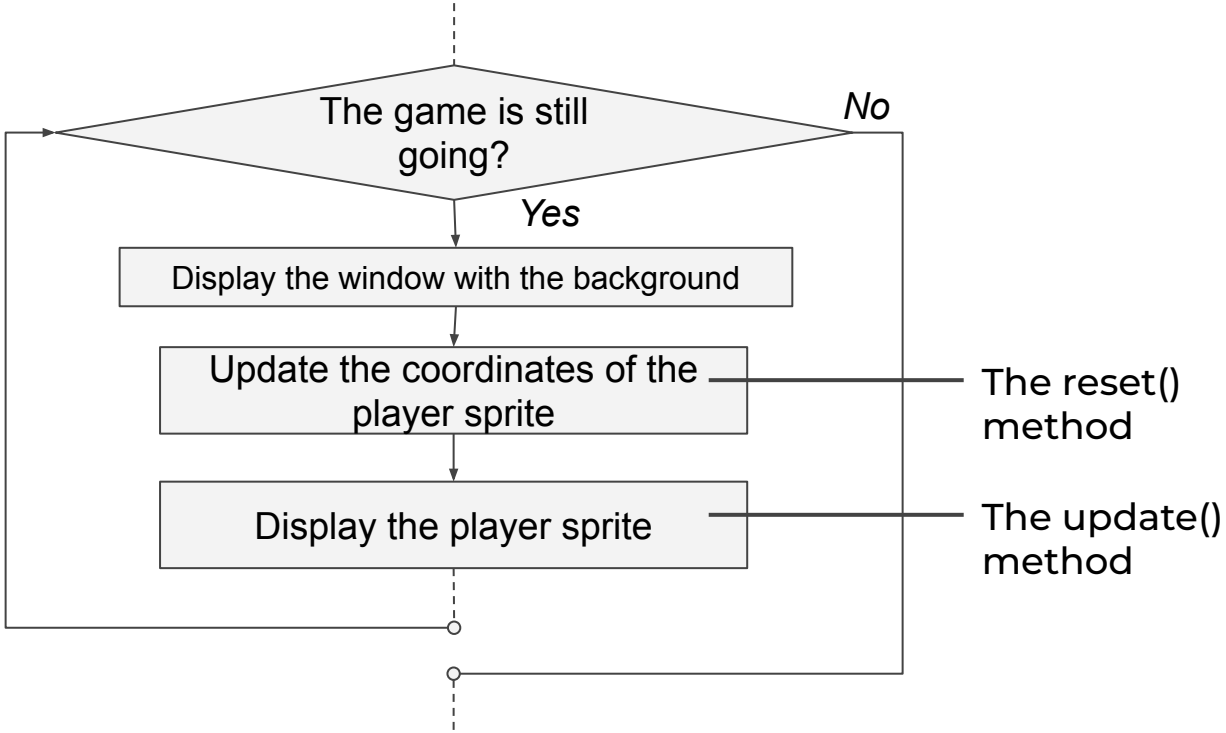
The GameSprite inheritor is supplemented only by the method of moving around the stage.

Brainstorming

The update() method — moving the sprite

The displaying of sprites is done in a game loop.

Let's define the update() method to change the coordinates of the sprite when the control keys are pressed.



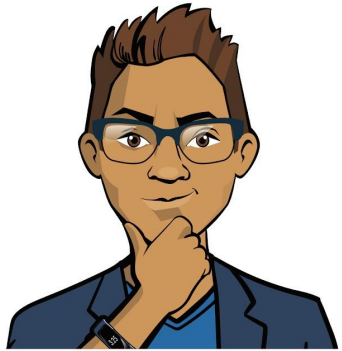
Brainstorming

The update() method — moving the sprite

The displaying of sprites is done in a game loop.

Let's define the update() method to change the coordinates of the sprite when the control keys are pressed.

How do we change the position of a sprite when an arrow key is pressed?



Brainstorming

The update() method — moving the sprite

The displaying of sprites is done in a game loop.

Let's define the update() method to change the coordinates of the sprite when the control keys are pressed.

Command	*Purpose*
`keys_pressed = key.get_pressed()`	Returns a structure with the current state of the keys (True — down, False — up).
`if keys_pressed[K_UP]:` `    y1 -= 10`	If the up arrow key is down, decrease the Y coordinate of Sprite1 by 10 pixels.
`if keys_pressed[K_s] and y2 < 395:` `    y2 += 10`	If the S key is down and the bottom of the screen has not been reached, increase the Y coordinate of Sprite2 by 10 pixels.

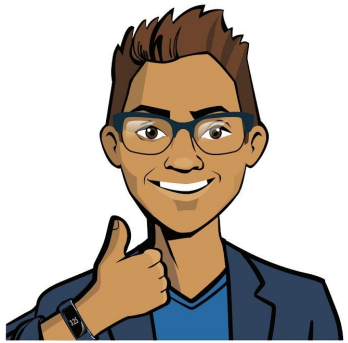
Brainstorming

The Player class

```
class Player(GameSprite):
    def update(self):
```

Controlling the sprite with the arrow keys

All the other properties and methods are already described in the GameSprite class!



Brainstorming

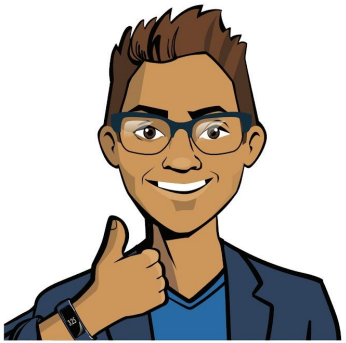
The Player class

```
class Player(GameSprite):
    def update(self):
```

Remember that the sprite should not go beyond the stage!

Controlling the sprite with the arrow keys

All the other properties and methods are already described in the GameSprite class!



Brainstorming

The Player class

```
class Player(GameSprite):
    def update(self):
        keys = key.get_pressed()
        if keys[K_LEFT] and self.rect.x > 5:
            self.rect.x -= self.speed
        if keys[K_RIGHT] and self.rect.x < win_width - 80:
            self.rect.x += self.speed
        if keys[K_UP] and self.rect.y > 5:
            self.rect.y -= self.speed
        if keys[K_DOWN] and self.rect.y < win_height - 80:
            self.rect.y += self.speed
```

Brainstorming

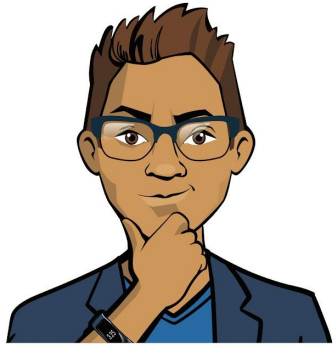
Let's introduce the Player class into the Maze game code

The result is a game stage with a main player that can be controlled.

- The GameSprite class description
- The Player class description
- Create background and connect music
- Creating sprites: *the player (instance of Player!), enemy,* and *treasure*
- Game loop:
  - Ending the game if the "Close window" button is pressed
  - Updating the location of the sprites
  - Rendering the stage and sprites on it
  - Updating the stage (next frame of the game loop)

Your tasks:

1. Implement the Player class.
2. Create an instance of the Player class for the player sprite.
3. Make changes to the game loop: at each step of the loop, the player's position must be updated.



Brainstorming

Module 5. Lesson 3. The Maze game. Part 2

Platform:

PyGame: Maze

Complete the tasks in VS Code

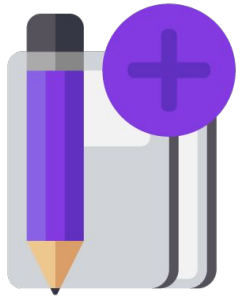
PyGame: Maze

Break

Module 5. Lesson 3. The Maze game. Part 2

Brainstorming:

The Enemy class



Let's program the Enemy class for the enemy guarding the treasure:

Checklist

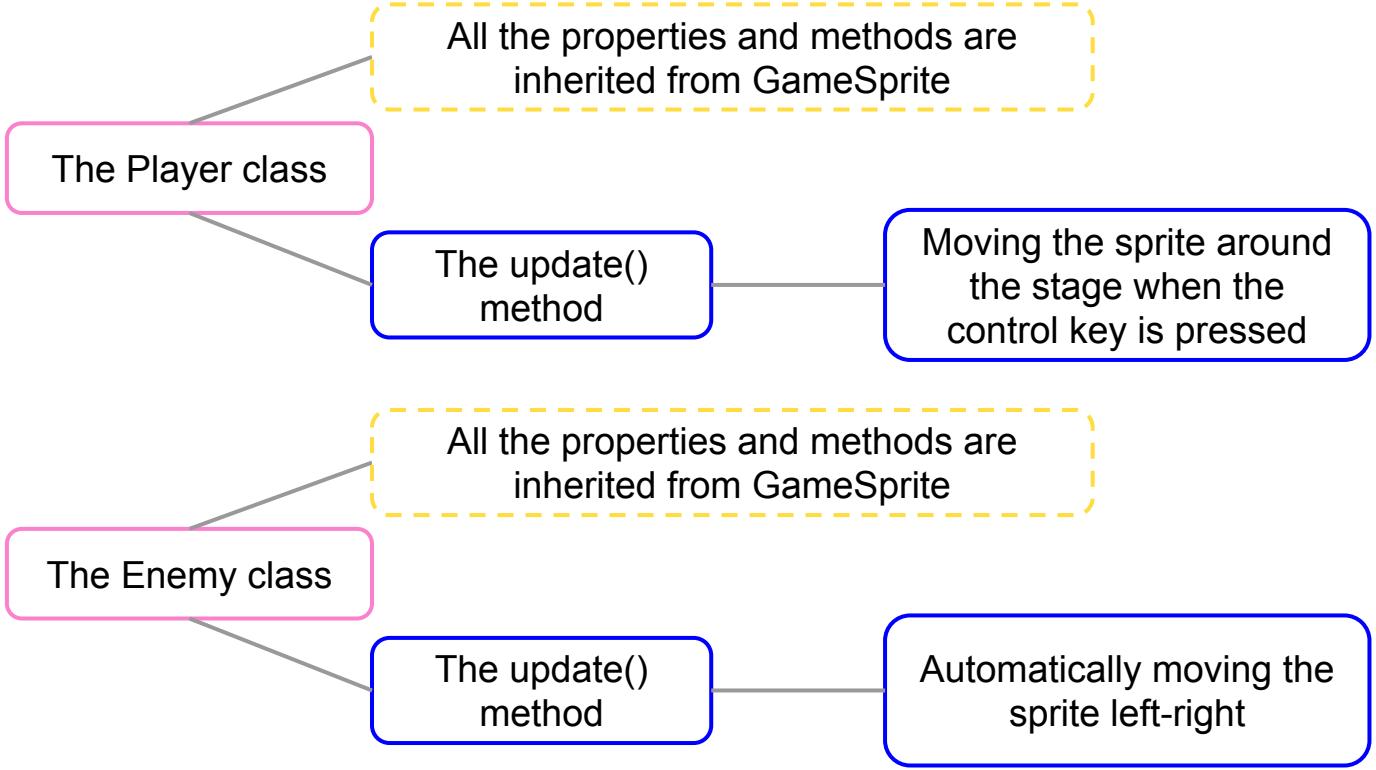
- [x] Create a game loop with an exit when you click on “Close Window”.
- [x] Set FPS ≈60 frames/sec
- [x] Set the background music
- [x] Create and display the player, enemy, and treasure sprites.
- [ ] Create a derived class for the player sprite
- [ ] Set key control for the player
- [ ] Create a derived class for the enemy sprite
- [ ] Set automatic movement for the enemy left-right (near the treasure)

Tasks for the second half of the work day

Brainstorming

The Player and Enemy classes

Let's create the update() method which is responsible for updating the position of the sprite on the stage.

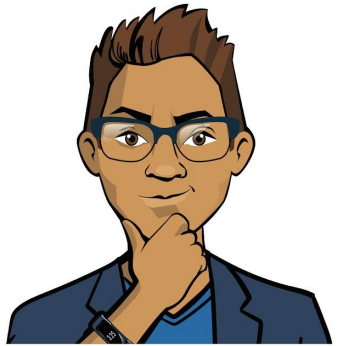


Brainstorming

The Enemy class

The Player class	The update() method	Moving the sprite around the stage when the control key is pressed
The Enemy class	The update() method	Automatically moving the sprite left-right

Will there be any errors due to the same names of the methods for Player and Enemy classes?



Brainstorming

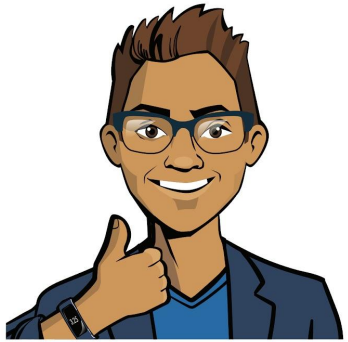
The Enemy class

The Player class	The update() method	Moving the sprite around the stage when the control key is pressed
The Enemy class	The update() method	Automatically moving the sprite left-right

`player.update()` → An instance of Player, calls update() from Player.

`monster.update()` → An instance of Enemy, calls update() from Enemy.

No! The method will be applied to a specific object, the class of which is known to the interpreter.



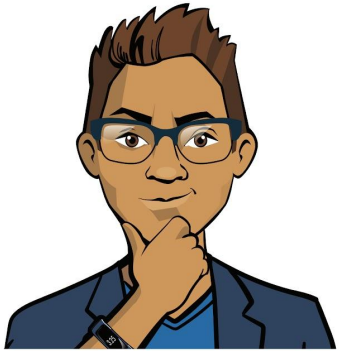
Brainstorming

The update() method — moving the enemy

The displaying of sprites is done in a game loop.

Let's define the update() method for automatic movement of the enemy.

How do we program automatic left-right movement between the points (x1, y) and (x2, y)?

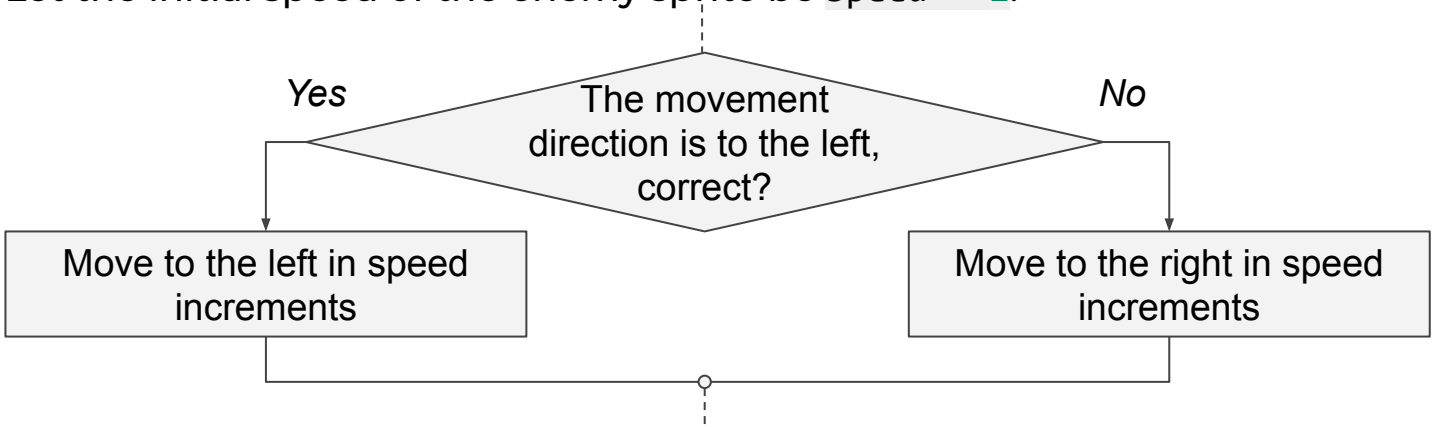


Brainstorming

The update() method — moving the enemy

Let's introduce in Enemy the property `direction = 'left'` — the direction of the movement.

Let the initial speed of the enemy sprite be `speed = 2`.



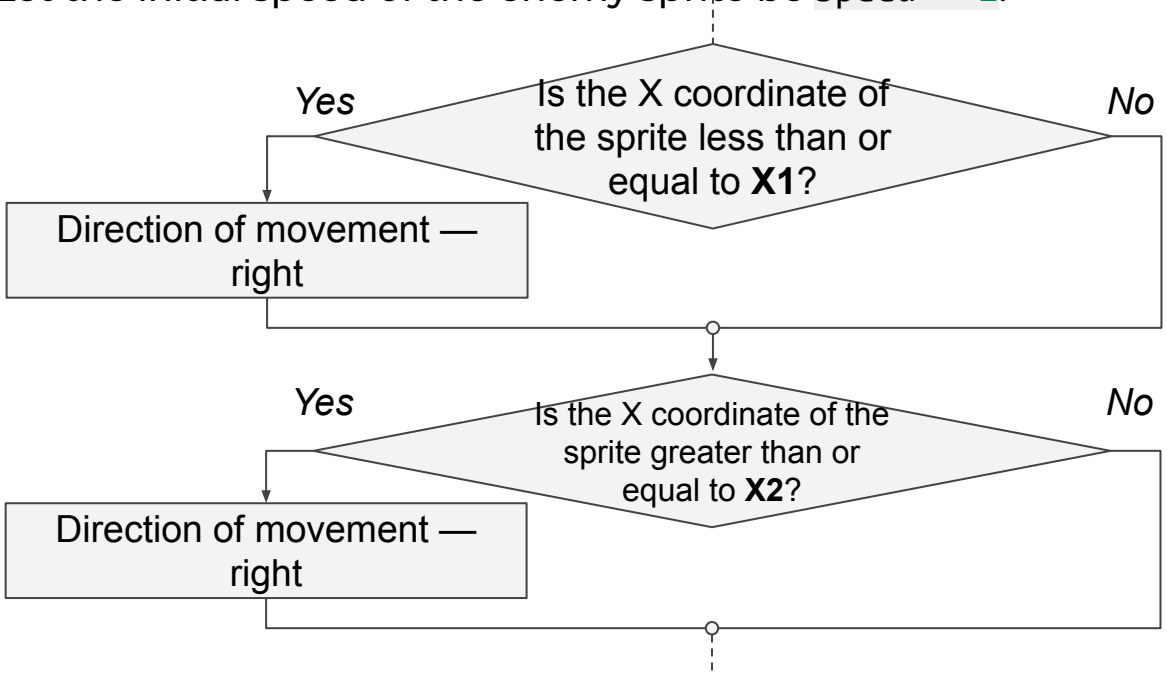
What if the leftmost point (x1, y) is reached?
How do we change the direction of movement (start moving towards (x2, y))?

Brainstorming

The update() method — moving the enemy

Let's introduce in Enemy the property `direction = 'left'` — the direction of the movement.

Let the initial speed of the enemy sprite be `speed = 2`.



Pick the coordinates of the boundary points (x1, y) and (x2, y) yourself.

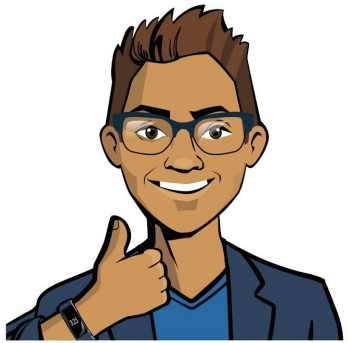
Brainstorming

The Enemy class

```
class Enemy(GameSprite):
    def update(self):
```

Automatic movement of the sprite left-right between two points

All the other properties and methods are already described in the GameSprite class!



Brainstorming

The Enemy class

```
class Enemy(GameSprite):
    def update(self):
        if self.rect.x <= 470:
            self.direction = "right"
        if self.rect.x >= win_width - 85:
            self.direction = "left"

        if self.direction == "left":
            self.rect.x -= self.speed
        else:
            self.rect.x += self.speed
```

Brainstorming

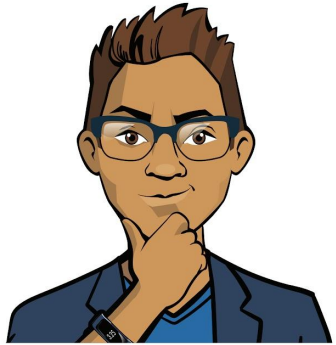
Let's introduce the Enemy class into the Maze game code

The result is a game stage with a main player and an enemy guarding the treasure.

- GameSprite class description
- The Player class description
- The Enemy class description
- Creating sprites: *the player, enemy (instance of Enemy)*, and *treasure*
- Game loop:
  - Ending the game if the "Close window" button is pressed
  - Updating the location of the sprites
  - Rendering the stage and sprites on it
  - Updating the stage (next frame of the game loop)

Your tasks:

1. Implement the Enemy class.
2. Create an instance of the Enemy class for the enemy sprite.
3. Make changes to the game loop: the enemy must move left-right, guarding the treasure.



Brainstorming

Module 5. Lesson 3. The Maze game. Part 2

Platform:

PyGame: Maze

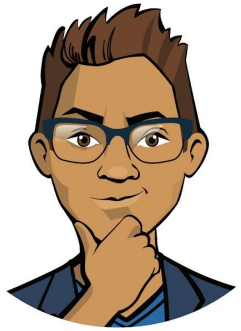
Complete the tasks in VS Code

PyGame: Maze

Wrapping up the work day

Let’s wrap up the work day by answering these technical questions:

1. Which classes did you work with today? How were they created?
2. Is it possible to create methods with the same name in different classes?



Cole,
senior developer

Emily,
project manager

Wrapping up the work day

Excellent work!

Colleagues,

You did a great job today.

On our next work day, we will complete the Maze game by creating obstacles and setting up the conditions for colliding with them!

Wrapping up the work day