Castle War

Tito Nicola Drugman & Vittoria Peppoloni

19/09/2022

CASTLE WAR

Code design

Create a well performing general structure, establish coding priorates

Use available resources

Learn some PyCharm shortcuts, create backup files and begin programming in a friendly environmental settings

Youtube, forums and docs

Find help online, ask questions to our colleagues or on forums. Use YouTube for the tutorials

Test the program

Run the program several times, check the performance

Debug

Fix errors, edit the program and understand some possible flaw

Final editing

Add more comments, create a user friendly and improve the design of the code

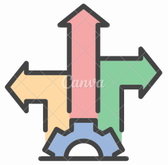
OUR IDEA

Establish priorities

Follow the schedule

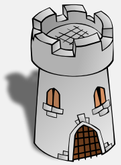
Follow step-by-step procedure

Choose the best path



Test the code

Ask friends to play the game



CASTLE WAR

Tito Nicola Drugman,
Vittoria Peppoloni

Part 1
The beginning

NEXT

Part 1: the beginning

Screen

Constants

Colors

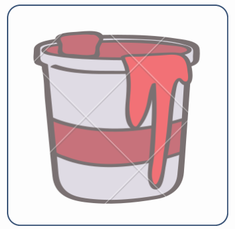
Font & icon

Create and display the screen

Import and use PyGame to create and display the screen

Import the constants

Copy and paste the constants available on Kiro. Add new ones (width and height of the units) edit some of them to personalize the code



Define the colors

Define all the colors required in the game by using RGB color code

Choose a font and an icon

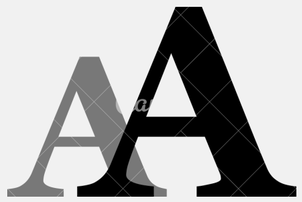
Select a proper font to use in the game and an icon for *Castle War*

Part 1: the beginning

Screen

Constants

Colors



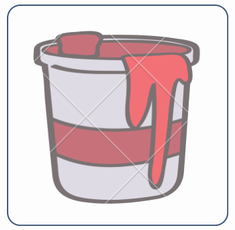
Font & icon

Create and display the screen

Import and use PyGame to create and display the screen

Import the constants

Copy and paste the constants available on Kiro. Add new ones (width and height of the units) edit some of them to personalize the code



Define the colors

Define all the colors required in the game by using RGB color code

Choose a font and an icon

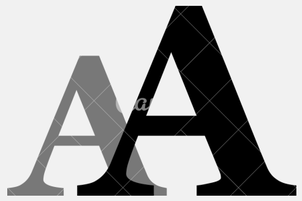
Select a proper font to use in the game and an icon for *Castle War*

Part 1: the beginning

Screen

Constants

Colors



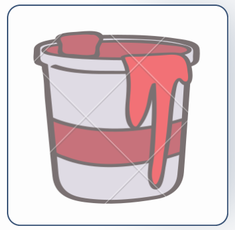
Font & icon

Create and display the screen

Import and use PyGame to create and display the screen

Import the constants

Copy and paste the constants available on Kiro. Add new ones (width and height of the units) edit some of them to personalize the code



Define the colors

Define all the colors required in the game by using RGB color code

Choose a font and an icon

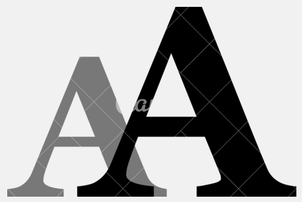
Select a proper font to use in the game and an icon for *Castle War*

Part 1: the beginning

Screen

Constants

Colors



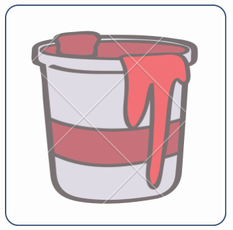
Font & icon

Create and display the screen

Import and use PyGame to create and display the screen

Import the constants

Copy and paste the constants available on Kiro. Add new ones (width and height of the units) edit some of them to personalize the code



Define the colors

Define all the colors required in the game by using RGB color code

Choose a font and an icon

Select a proper font to use in the game and an icon for *Castle War*

Tito Nicola Drugman,
Vittoria Peppoloni

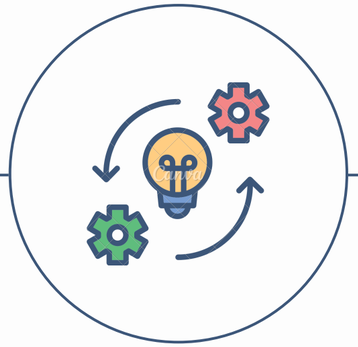
Part 2
Create classes

NEXT

The strength of classes and why we used them

Easy

Well-readable and easy to create, allows to control multiple objects at the same time



More with less

Create multiple objects with similar property. Helpful in case of alike units

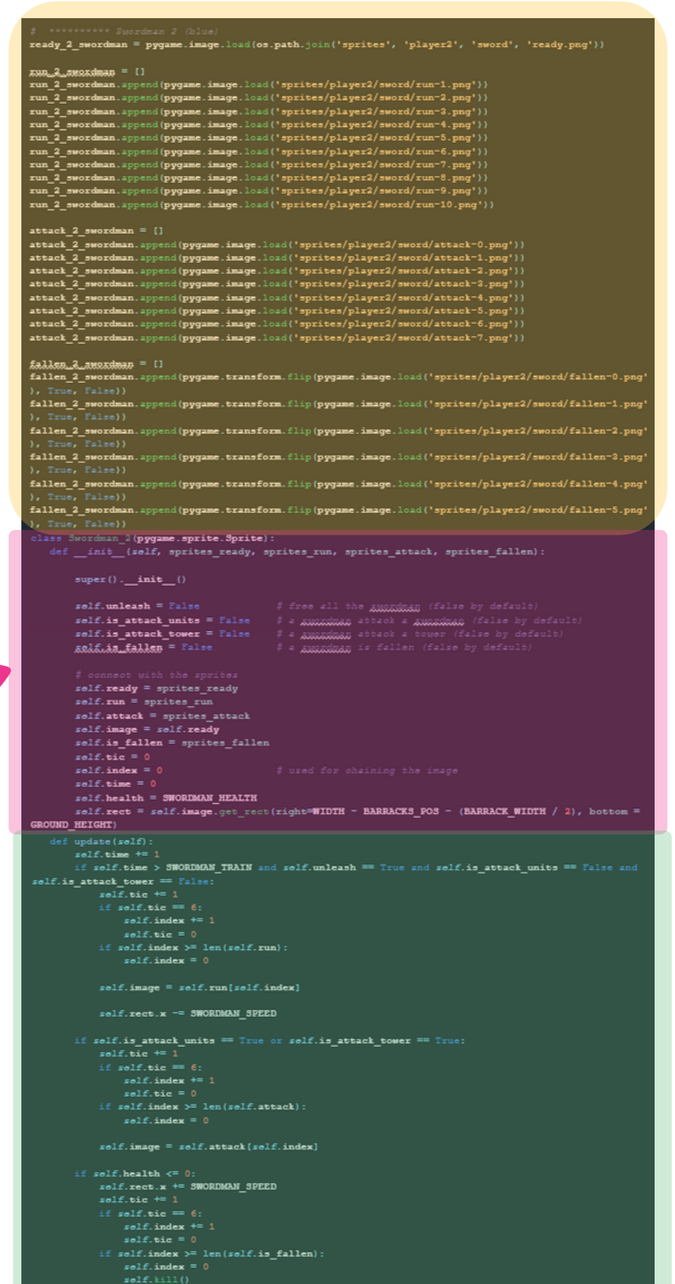
Copy and paste

Easy to adapt, just copy and paste and with little edit to design a new class

Use methods

Use methods in classes to enable objects to change their properties

Example: class Swordman 2



```
# ********** Swordman 2 (blue)
ready_2_swordman = pygame.image.load(os.path.join('sprites', 'player2', 'sword', 'ready.png'))

run_2_swordman = []
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-1.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-2.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-3.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-4.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-5.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-6.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-7.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-8.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-9.png'))
run_2_swordman.append(pygame.image.load('sprites/player2/sword/run-10.png'))

attack_2_swordman = []
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-0.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-1.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-2.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-3.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-4.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-5.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-6.png'))
attack_2_swordman.append(pygame.image.load('sprites/player2/sword/attack-7.png'))

fallen_2_swordman = []
fallen_2_swordman.append(pygame.transform.flip(pygame.image.load('sprites/player2/sword/fallen-0.png'
), True, False))
fallen_2_swordman.append(pygame.transform.flip(pygame.image.load('sprites/player2/sword/fallen-1.png'
), True, False))
fallen_2_swordman.append(pygame.transform.flip(pygame.image.load('sprites/player2/sword/fallen-2.png'
), True, False))
fallen_2_swordman.append(pygame.transform.flip(pygame.image.load('sprites/player2/sword/fallen-3.png'
), True, False))
fallen_2_swordman.append(pygame.transform.flip(pygame.image.load('sprites/player2/sword/fallen-4.png'
), True, False))
fallen_2_swordman.append(pygame.transform.flip(pygame.image.load('sprites/player2/sword/fallen-5.png'
), True, False))
```

```
class Swordman_2(pygame.sprite.Sprite):
    def __init__(self, sprites_ready, sprites_run, sprites_attack, sprites_fallen):

        super().__init__()

        self.unleash = False            # frees all the swordman (false by default)
        self.is_attack_units = False    # a swordman attack a swordman (false by default)
        self.is_attack_tower = False    # a swordman attack a tower (false by default)
        self.is_fallen = False          # a swordman is fallen (false by default)

        # connect with the sprites
        self.ready = sprites_ready
        self.run = sprites_run
        self.attack = sprites_attack
        self.image = self.ready
        self.is_fallen = sprites_fallen
        self.tic = 0
        self.index = 0                  # used for chaining the image
        self.time = 0
        self.health = SWORDMAN_HEALTH
        self.rect = self.image.get_rect(right=WIDTH - BARRACKS_POS - (BARRACK_WIDTH / 2), bottom =
GROUND_HEIGHT)
```

```
    def update(self):
        self.time += 1
        if self.time > SWORDMAN_TRAIN and self.unleash == True and self.is_attack_units == False and
self.is_attack_tower == False:
            self.tic += 1
            if self.tic == 6:
                self.index += 1
                self.tic = 0
            if self.index >= len(self.run):
                self.index = 0

            self.image = self.run[self.index]

            self.rect.x -= SWORDMAN_SPEED

        if self.is_attack_units == True or self.is_attack_tower == True:
            self.tic += 1
            if self.tic == 6:
                self.index += 1
                self.tic = 0
            if self.index >= len(self.attack):
                self.index = 0

            self.image = self.attack[self.index]

        if self.health <= 0:
            self.rect.x += SWORDMAN_SPEED
            self.tic += 1
            if self.tic == 6:
                self.index += 1
                self.tic = 0
            if self.index >= len(self.is_fallen):
                self.index = 0
                self.kill()
```

List and sprites

Create an empty list for each type of sprite (*ready*, *run*, *attack*, *fallen*) then add the sprites

Define and initialize

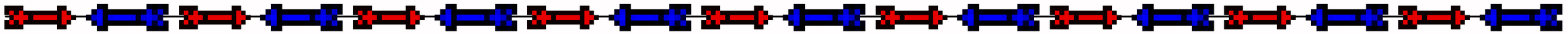
Define and initialize a class with the *def __init__* method

def update(self)

Check condition to move from one state to another (*ready*, *run*, *attack*, *fallen*)

```
    def update(self):
        self.time += 1
        if self.time > SWORDMAN_TRAIN and self.unleash == True and
self.is_attack_units == False and self.is_attack_tower == False:
            self.tic += 1
            if self.tic == 6:
                self.index += 1
                self.tic = 0
            if self.index >= len(self.run):
                self.index = 0

            self.image = self.run[self.index]

            self.rect.x += SWORDMAN_SPEED


        if self.is_attack_units == True or self.is_attack_tower == True:
            self.tic += 1
            if self.tic == 6:
                self.index += 1
                self.tic = 0
            if self.index >= len(self.attack):
                self.index = 0

            self.image = self.attack[self.index]

        if self.health <= 0:
            self.rect.x -= SWORDMAN_SPEED
            self.tic += 1
            if self.tic == 6:
                self.index += 1
```

A problem: tower's arrows



01

What is The Problem?

Enable the tower to shoot towards multiple targets

02

Which is the solution?

Design multiple ranges

Solution

Create two ranges

✓ **RANGE 1**

The tower shoots the first arrow when one of the enemy units enters the range

✓ **RANGE 2**

But it also shoots another arrow when the same enemy unit enters the second range

✓ **THE SOLUTION**

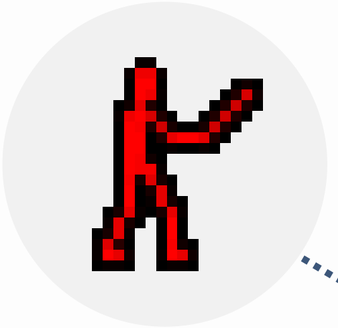
It is more visual appealing to have the tower shooting in two directions, also very easy to develop

Tito Nicola Drugman,
Vittoria Peppoloni

Part 3
Units

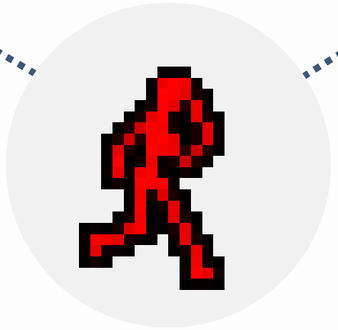
NEXT

LIFE (and death) OF A SWORDMAN



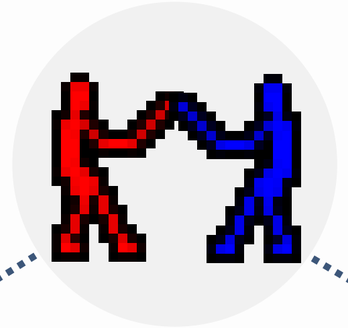
Generate a swordman

Assuming you have the necessary resources, by pressing W for PL1 or O for PL2 a swordman will be generated.



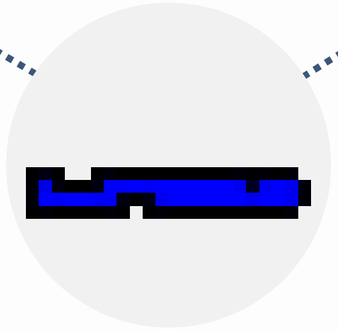
Start running

Pressing D for PL1 or J for PL2 will enable swordman running.



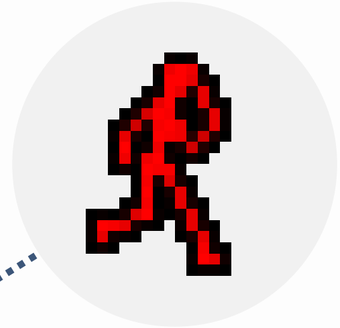
Duel

As soon as an unit runs into an enemy unit from the opposite tower, the fight begins. There can be only one winner.



Fallen swordman

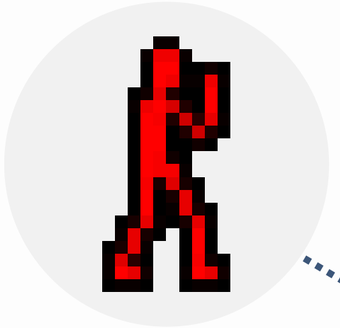
If the unit health is less or equal to 0 then it is consider to be death.



Start running again

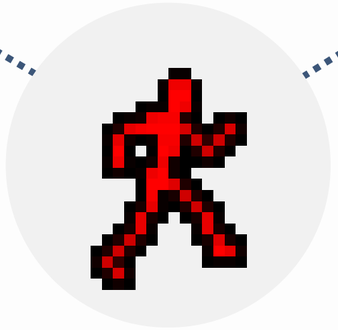
When the duel is over the winning swordman will start running again

LIFE (and life) OF A WORKER



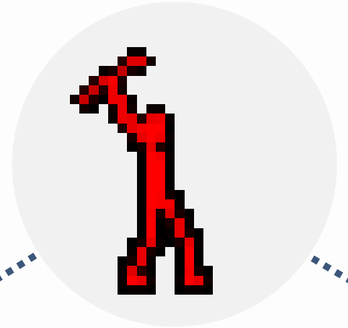
Generate a worker

Assuming the needed resources are available, by pressing Q for PL1 or P for PL2 you can generate a worker.



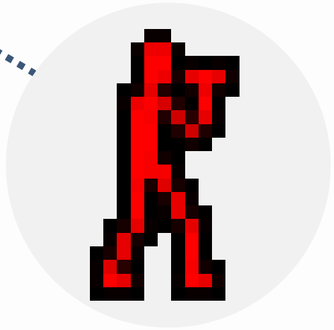
Start running

When the worker is ready it can be deployed.



Go to mine...

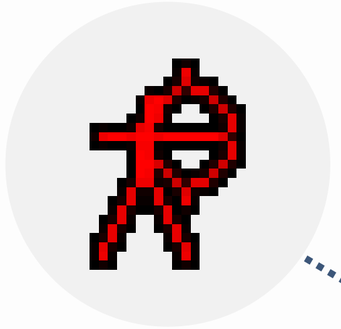
By pressing A for PL1 or L for PL2 the worker can reach the mine and begin collecting resources.



... or go to wall

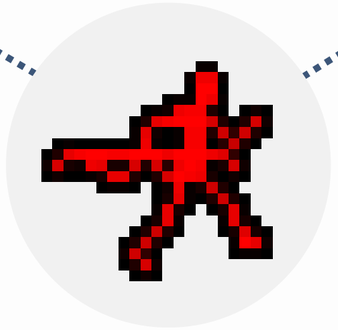
By pressing S for PL1 or K for PL2 the worker can move to the wall and start healing it back.

LIFE (and death) OF A BOWMAN



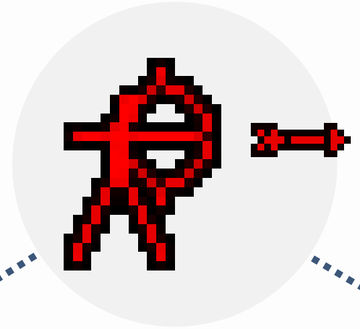
Generate a bowman

By pressing E for PL1 or I for PL2 if you have the necessary resources you can generate a bowman.



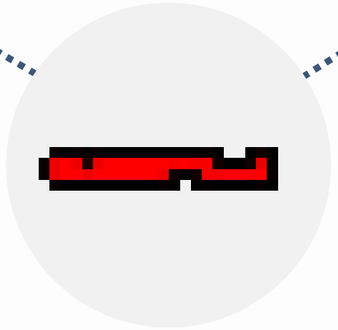
Start running

By pressing F for PL1 or H for PL2 all the bowman start running.



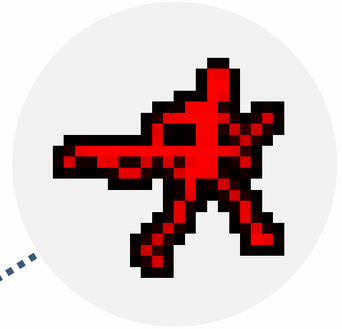
Start shooting

As soon an enemy unit is in the range the bowman stops running and starts shooting.



Fallen bowman

If the health of the unit is less or equal to 0 then it is consider death.

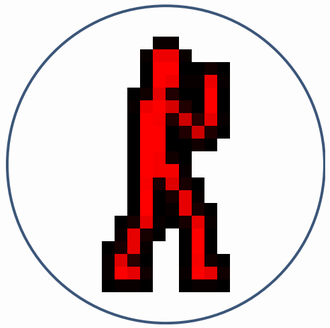


Start running again

After a duel if the bowman is still alive it starts running again.

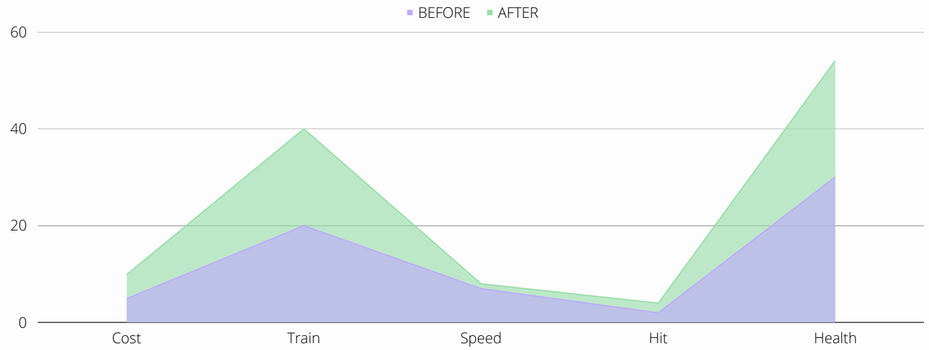
Personal changes

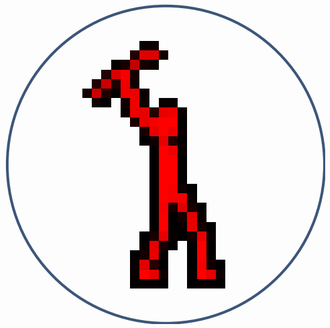
Graphs



Swordman

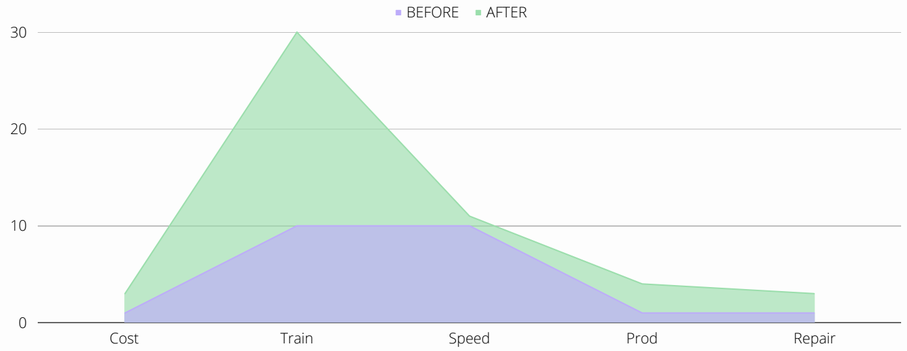
Decrease swordsman health from 30 to 24.
Decrease swordman speed from 7 to 1.

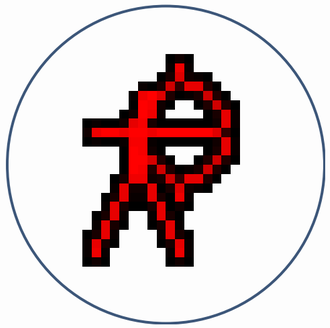




Worker

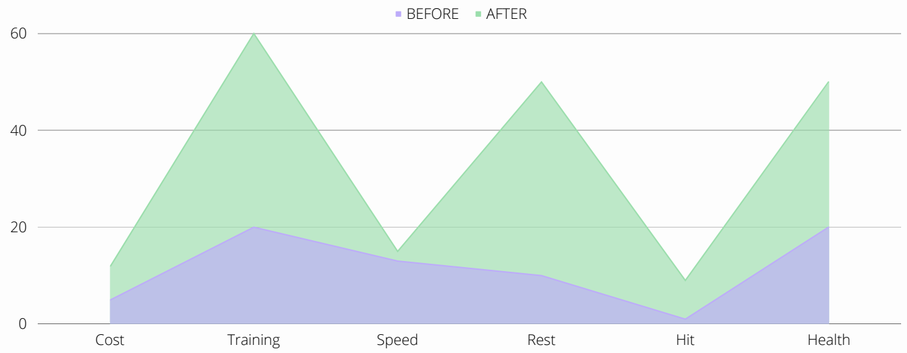
Worker cost, worker train time and worker repair doubled. Worker speed reduced from 10 to 1. Production triplicated from 1 to 3.





Bowman

Cost increased from 5 to 7, train doubled from 20 to 40 and speed reduced from 13 to 2. Resting time quadrupled and damaged caused by arrow increased from 1 to 8. Heath increased from 20 to 30.



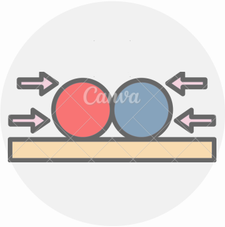
How to manage units

```
for arrow in Arrow_1_list:
    if pygame.sprite.spritecollideany(bowman, Arrow_1_list):
        Arrow_1_list.remove(arrow)
        for bowman in Bowman_2_list:
            if bowman.isshoot == True:
                bowman.health -= BOWMAN_HIT
```

Easy to learn on PyGame

Easy to develop

pygame.sprite.spritecollideany



Check collision with a sprite of a group

If it is true perform an action

Copy and paste

If it is false keep checking

Tito Nicola Drugman,
Vittoria Peppoloni

Part 4
Adding extra

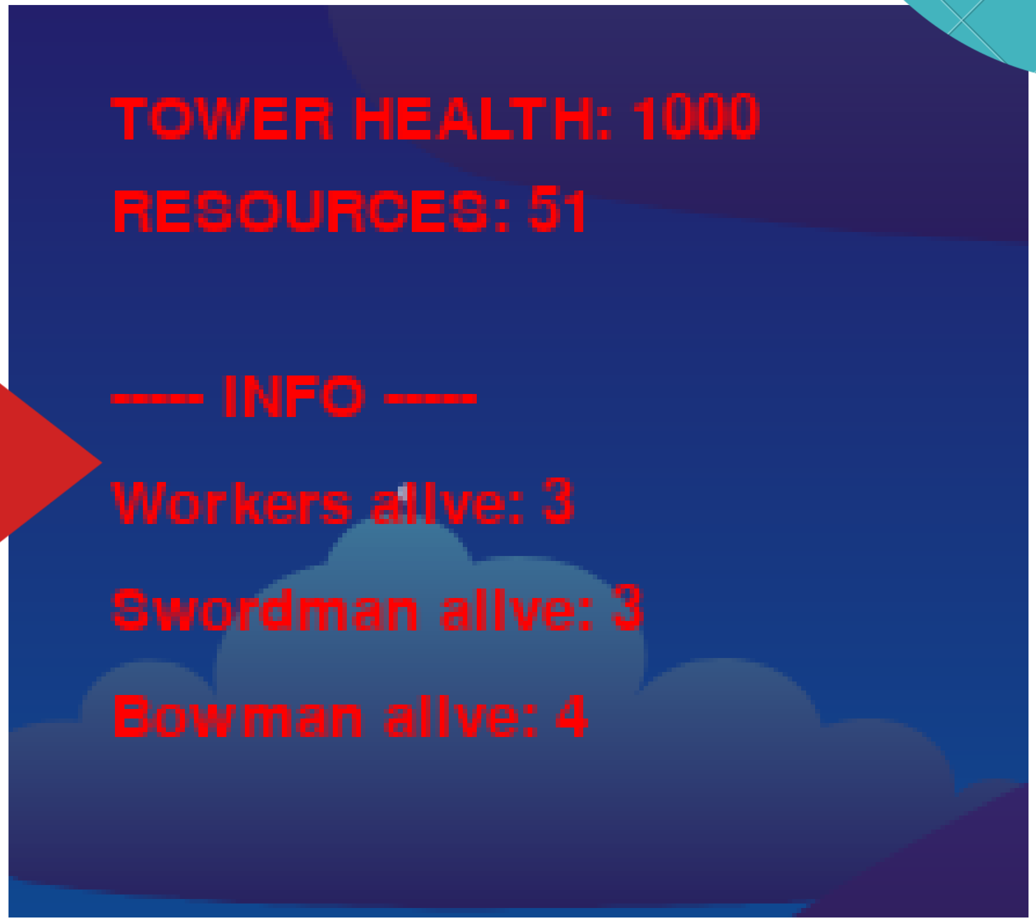
NEXT

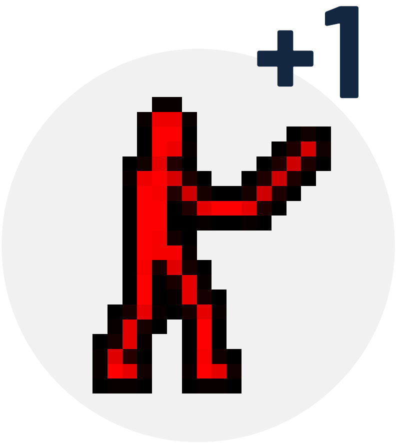
Display info

ADDING EXTRA

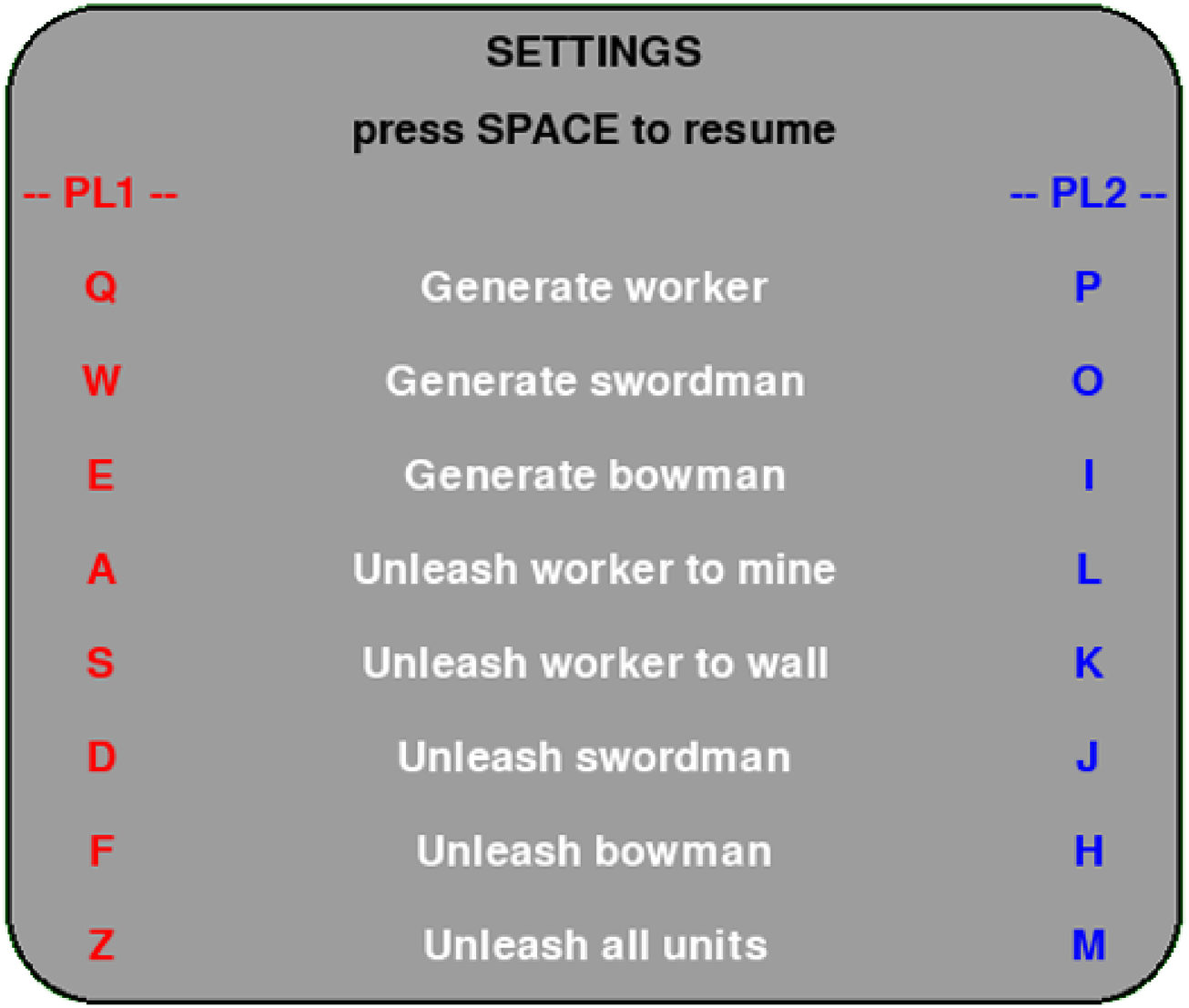
Keep the count of all the units alive

```
# \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
# Player 1 units

redinfo_text = font_number2.render("——— INFO ———", True, "RED")
redinfo_text_rect = redinfo_text.get_rect()
redinfo_text_rect.topleft = (font_distance_x, temp_y)

red_swordman_text = font_number2.render("Swordman alive: " + str(len(Swordman_1_list)), True, "
red_swordman_text_rect = red_swordman_text.get_rect()
red_swordman_text_rect.topleft = (font_distance_x, temp_y + (font_distance_y * 2))

red_worker_text = font_number2.render("Workers alive: " + str(len(Worker_1_list)), True, "RED"
red_worker_text_rect = red_worker_text.get_rect()
red_worker_text_rect.topleft = (font_distance_x, temp_y + (font_distance_y * 1))

red_bowman_text = font_number2.render("Bowman alive: " + str(len(Bowman_1_list)), True, "RED")
red_bowman_text_rect = red_bowman_text.get_rect()
red_bowman_text_rect.topleft = (font_distance_x, temp_y + (font_distance_y * 3))
```





ADDING EXTRA



Design a setting tab

01 Before the wallpaper was settled we started with only a gray rectangle and add on the corner 4 circles to make it rounded

02 We decided to display all the elements one by one, so that we could edit them individually

03 Use *rect.center* to display the text precisely and edit the size and color

04 When the wallpaper was added we first created the border, to fill it inside we duplicated the border, change the color and increase enormously the thickness

Display stats

ADDING EXTRA

01 - Rectangles

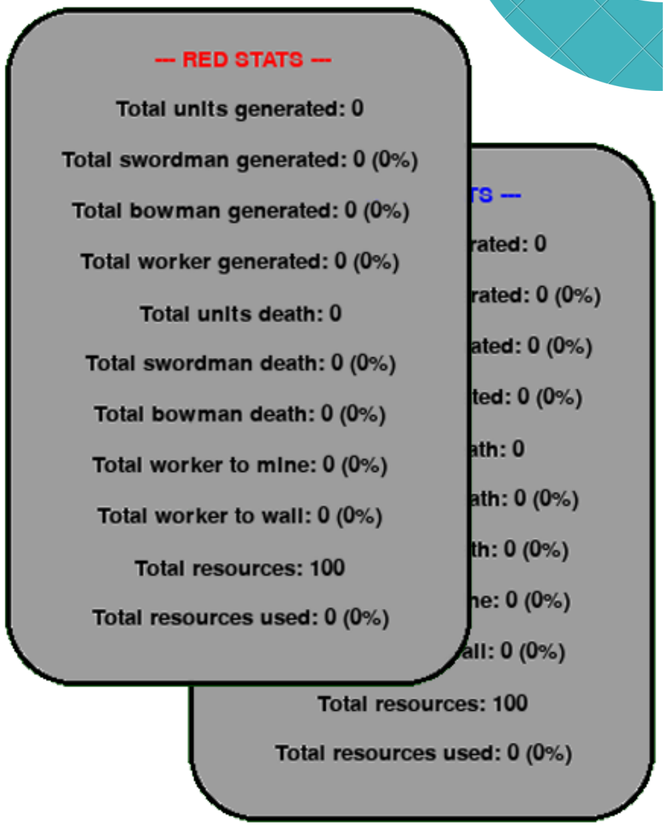
Create rounded-corners rectangles

02 - Variables

Create global variables that can be increased easly

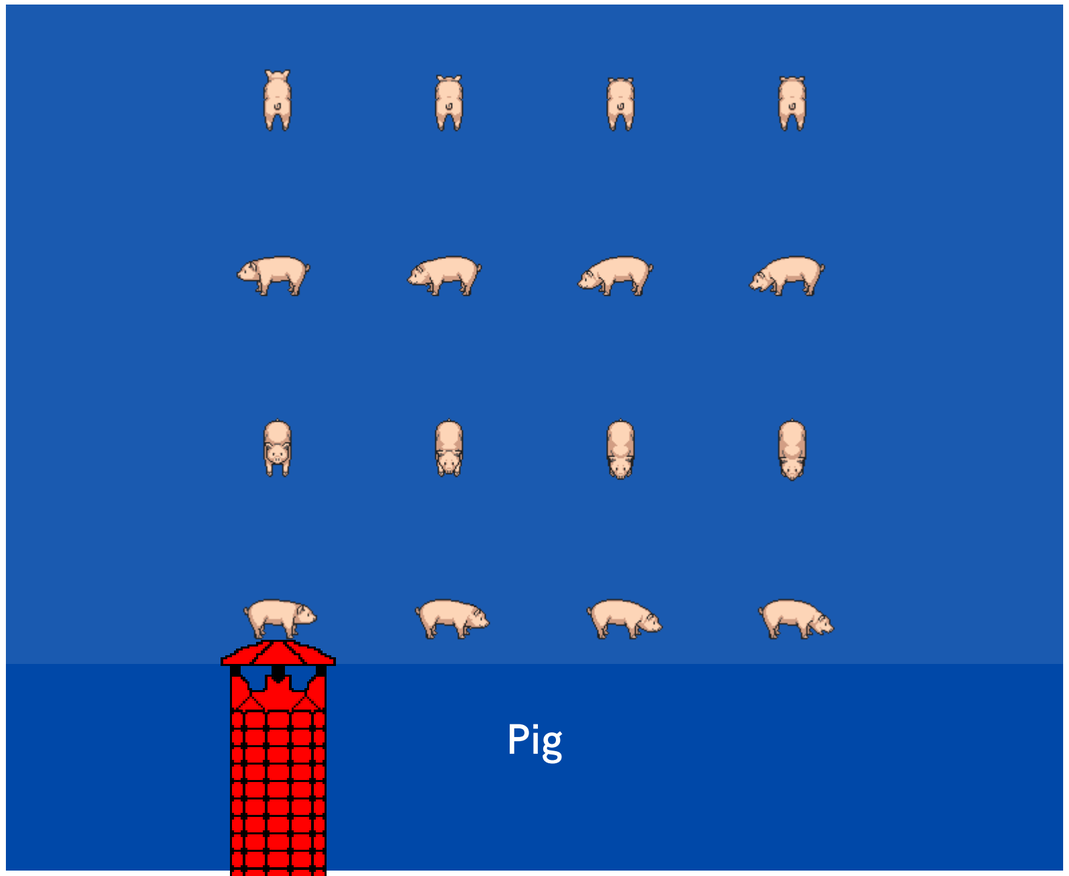
02 - Divide by 0

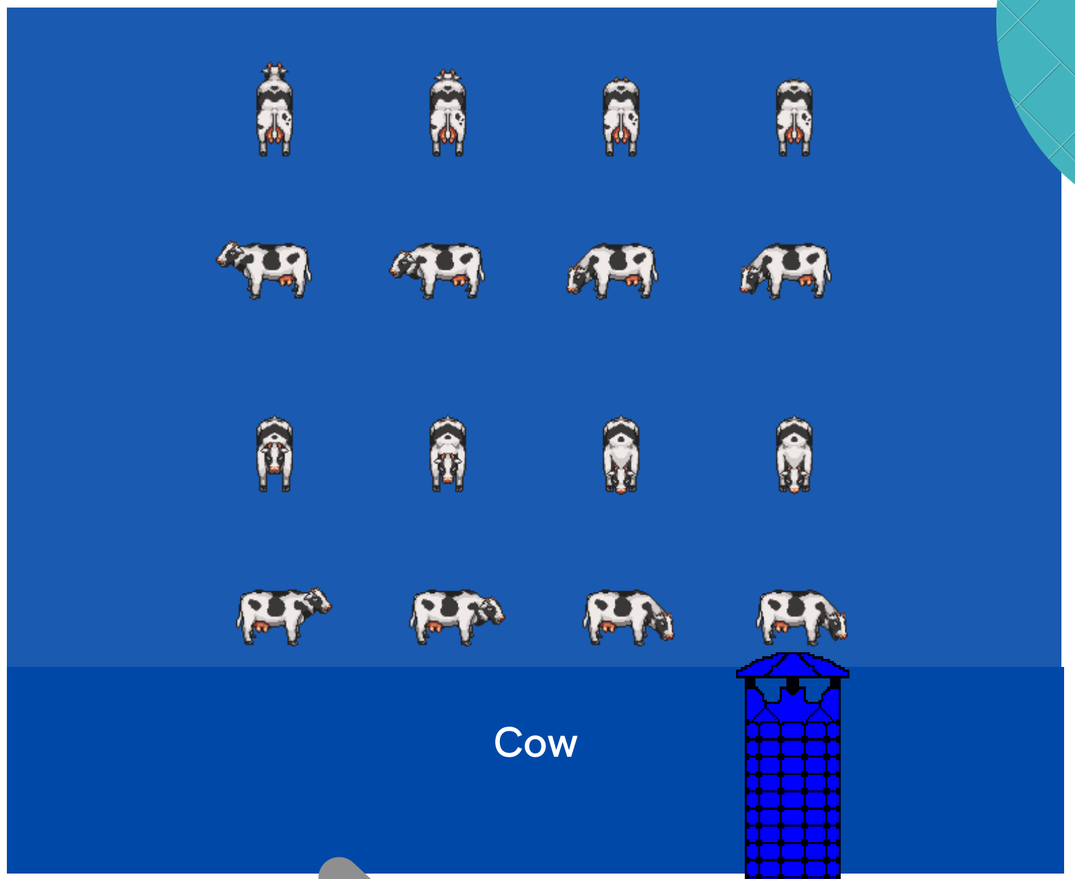
Use *try* and *except* to avoid dividing by 0 when calculating the percentages



Next Page

Adding animals





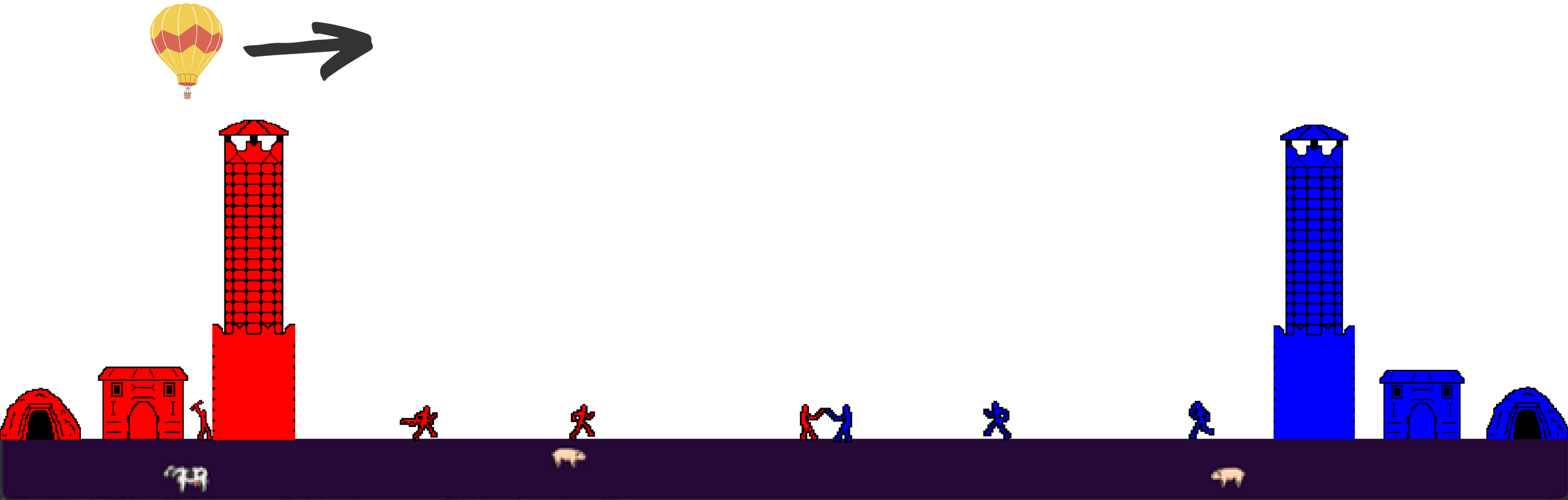
Tito Nicola Drugman,
Vittoria Peppoloni

Part 5
Improvements

NEXT

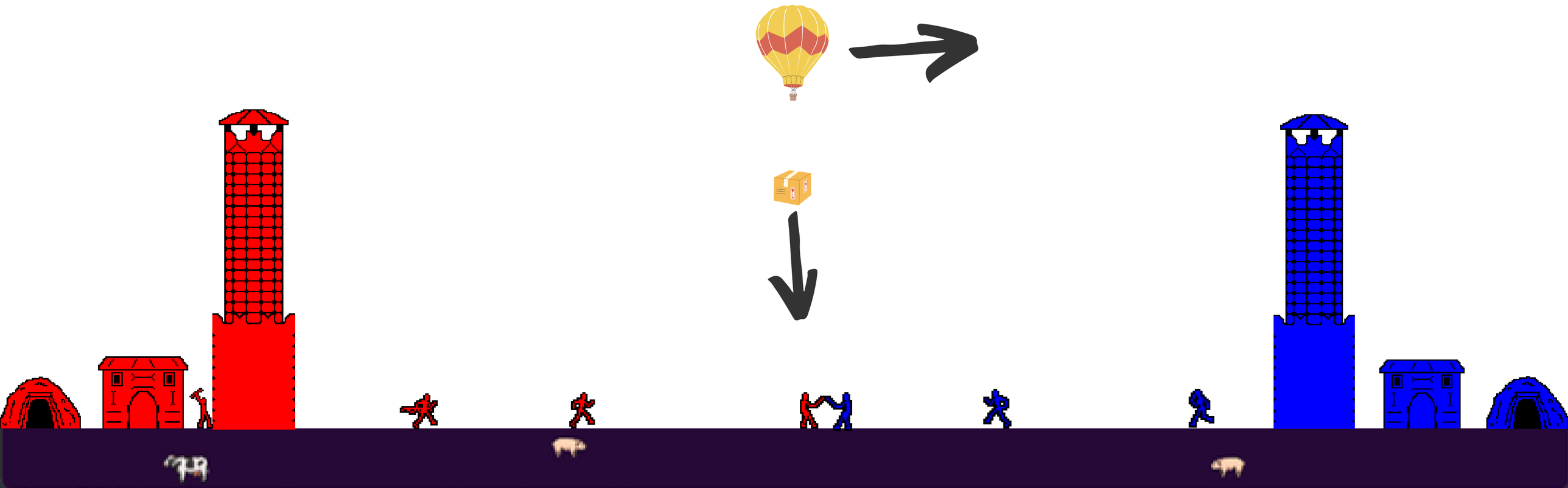
Hot air balloon

Randomly when it reaches the center of the screen...



Hot air balloon

... it drops a bonus



Hot air balloon

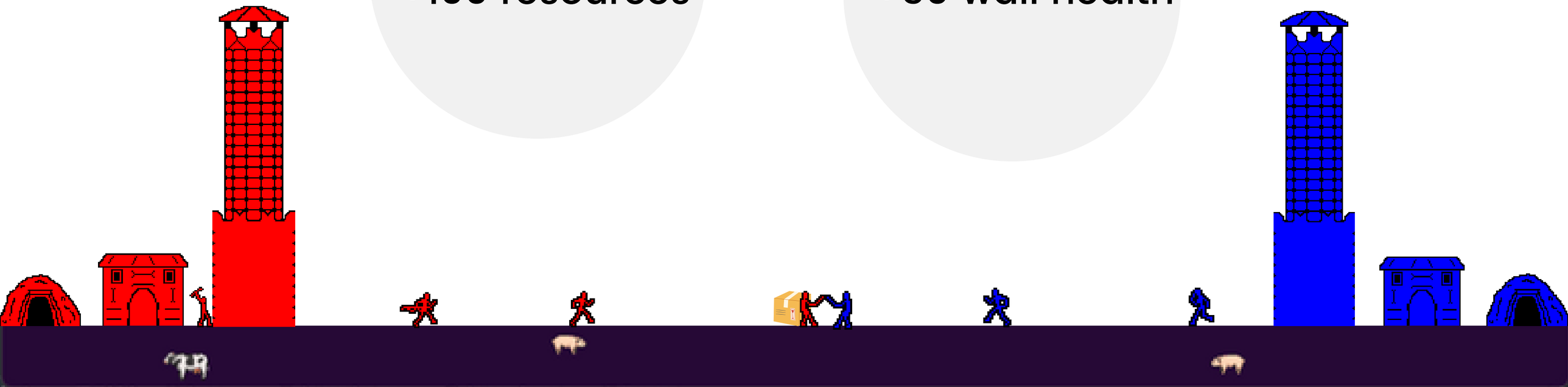
Can be one of those two things

50%

+100 resources

50%

+50 wall health



QUESTIONS?

LET US KNOW IF YOU
HAVE ANY QUESTIONS

Tito Nicola Drugman,
Vittoria Peppoloni

Thank you!