### 

## **Задачи**

### **1. Стратегическая компьютерная игра**

В стратегической компьютерной игре игроки сражаются с друг другом, добывая ресурсы и строя армии и здания.

Игроку должно даваться на выбор несколько различных наборов (рас, кланов и т.п.) боевых единиц и зданий. Боевые единицы и здания должны иметь следующие характеристики:

* стоимость - количество ресуров, необходимых для постройки здания или воспроизведения боевой единицы,
* очки здоровья - уменьшаются при нанесении урона (если они заканчиваются, то соответствующее здание или боевая единица уничтожается),
* очки энергии - уменьшаются при использовании специальных навыков,
* показатель атаки - определяет размер урона,
* показатель защиты - уменьшает наносимый урон.

При желании вы можете расширить набор характеристик.

Тематику игры выберите самостоятельно.

#### **Ресурсы**

Экономика игры должна состоять из нескольких ресурсов.

Особенности ресурсов:

* набор ресурсов одинаков для всех игроков вне зависимости от выбранного набора боевых единиц и зданий.
* воспроизводство ресурсов происходит с помощью специальных зданий,
* игрок тратит ресурсы на покупку боевых единиц и постройку зданий.

Перечень ресурсов и их особенности определите самостоятельно.

#### **Здания**

Здания - основа экономики игрока.

Типы зданий:

* воспроизводящие ресурсы,
* воспроизвощящие боевые единицы.

Перечень зданий, их характеристики и особенности определите самостоятельно.

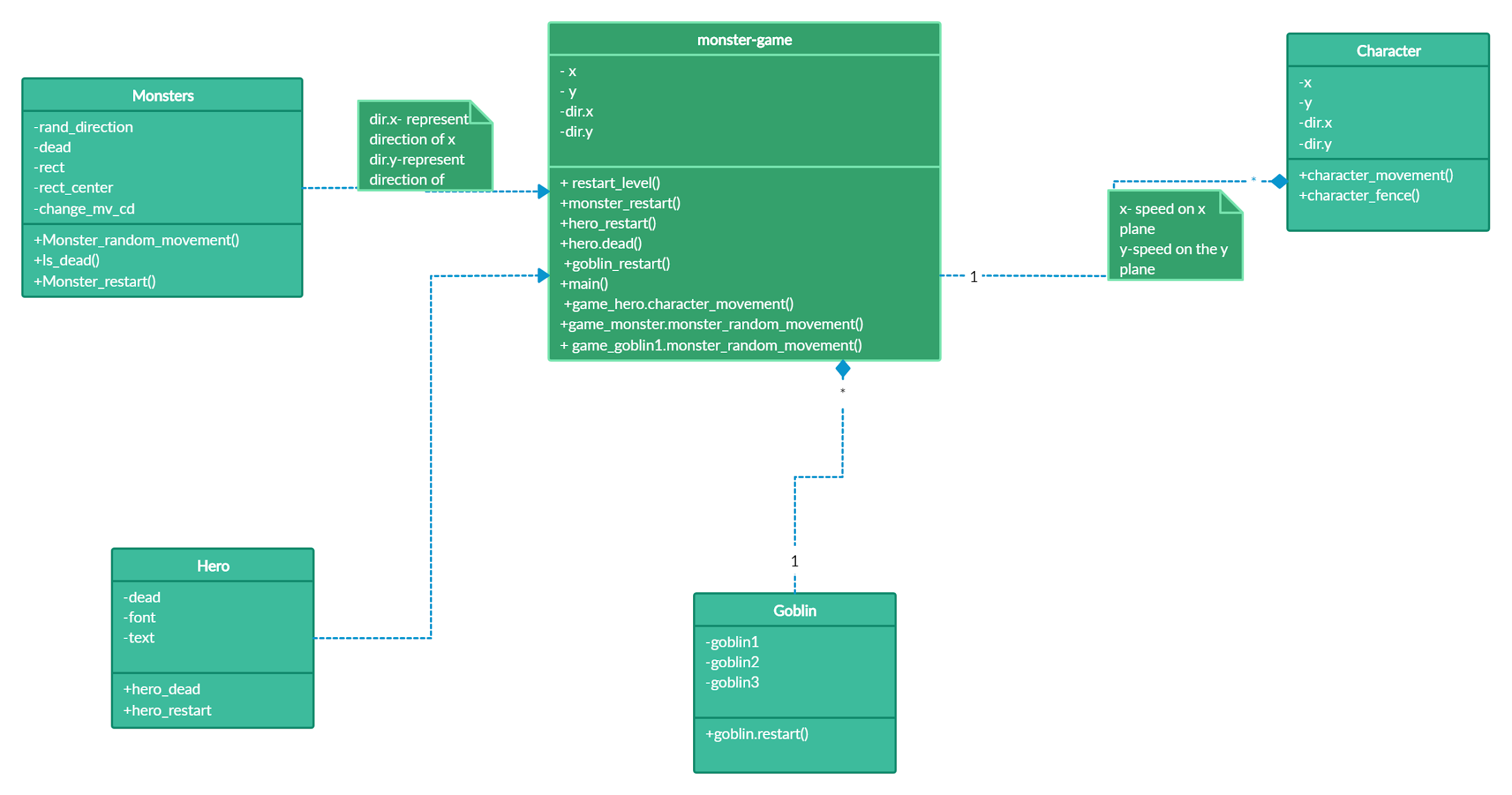
#### **Боевые единицы**

Боевые единицы - основа армии игрока.

Особенности боевых единиц:

* воспроизводятся в специальных зданиях,
* может атаковать другие боевые единицы и здания, уменьшая у них при этом количество очков здоровья,
* одна боевая единица может иметь специальные навыки, на использование которых тратятся очки энергии,
* несколько боевых единиц можно объединять в отряды.

Перечень боевых единиц, их характеристики и особенности определите самостоятельно.



**Исходный код**

*characters.py*

import pygame

class Characters(pygame.sprite.Sprite):

# Initialize with x, y position as well as x,y speed

def \_\_init\_\_(self, x, y, dir\_x, dir\_y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.x = x

self.y = y

self.dir\_x = dir\_x

self.dir\_y = dir\_y

self.dead = False

# Function that controls the character's movement

def character\_movement(self):

self.x += self.dir\_x

self.y += self.dir\_y

# Function that controls the character's boundaries

def character\_fence(self, option):

# Hero Fence is Option 1

if option == 1:

if self.x + self.dir\_x > 450:

self.x = 450

if self.x + self.dir\_x < 33:

self.x = 33

if self.y + self.dir\_y > 415:

self.y = 415

if self.y + self.dir\_y < 33:

self.y = 33

# Monster Fence is Option 2

if option == 2:

if self.x + self.dir\_x > 450: # If the Monster's next move is past the trees (RIGHT), go to the opposite direction

self.x = 33

if self.x + self.dir\_x < 33: # If the Monster's next move is past the trees (LEFT), go to the opposite direction

self.x = 450

if self.y + self.dir\_y > 415: # If the Monster's next move is past the trees (SOUTH), go to the opposite direction

self.y = 33

if self.y + self.dir\_y < 33: # If the Monster's next move is past the trees (NORTH), go to the opposite direction

self.y = 415

def rect\_update(self):

self.rect.center = self.x, self.y

*hero.py*

import pygame

from characters import Characters

class Hero(Characters):

# Import Hero image

hero\_image = pygame.image.load("./images/hero.png")

# Initialize with the location of the hero

def \_\_init\_\_(self, x, y):

Characters.\_\_init\_\_(self, x, y, 0, 0)

self.image = pygame.image.load("./images/hero.png").convert\_alpha()

self.rect = self.image.get\_rect()

self.rect.center = self.x, self.y

def hero\_dead(self):

self.dead = True

font = pygame.font.Font(None, 60)

text = font.render("HA! You Lose!", True, (0, 0, 0))

self.image = text

self.x = 120

self.y = 150

self.dir\_x = 0

self.dir\_y = 0

def hero\_restart(self):

self.image = pygame.image.load("./images/hero.png").convert\_alpha()

self.x = 256

self.y = 240

self.dir\_x = 0

self.dir\_y = 0

self.dead = False

*Monster.py*

import random

import sys

# from PyQt5.QtWidgets import QtGui, QtCore

import pygame

from characters import Characters

class Monster(Characters):

# Import Monster image

monster\_image = pygame.image.load("./images/monster.png")

# Initialize wih the starting speed and direction of the monster

def \_\_init\_\_(self, x, y, dir\_x, dir\_y):

Characters.\_\_init\_\_(self, x, y, dir\_x, dir\_y)

self.change\_mv\_cd = 60

self.image = pygame.image.load("./images/monster.png").convert\_alpha()

self.rect = self.image.get\_rect()

self.rect.center = self.x, self.y

# Function countdowns the time until it has to randomize the movement.

# When countdown is 0, it will randomly change direction

def monster\_random\_movement(self):

self.change\_mv\_cd -= 1

if self.change\_mv\_cd == 0:

self.change\_mv\_cd = 60

rand\_direciton = random.randint(0, 3)

if rand\_direciton == 0: # Go North

self.dir\_y = -self.dir\_y

if rand\_direciton == 1: # Go Right

self.dir\_x = -self.dir\_x

if rand\_direciton == 2: # Go South

self.dir\_y = -self.dir\_y

if rand\_direciton == 3: # Go Left

self.dir\_x = -self.dir\_x

# Function that updates the monster's attributes after collision

def is\_dead(self):

self.dead = True

font = pygame.font.Font(None, 60)

text = font.render("You Win!", True, (0, 0, 0))

self.image = text

self.x = 172

self.y = 150

self.dir\_x = 0

self.dir\_y = 0

def monster\_restart(self):

self.image = pygame.image.load("./images/monster.png").convert\_alpha()

self.x = random.randint(40, 440)

self.y = random.randint(40, 400)

self.dir\_x = 3

self.dir\_y = 3

self.dead = False

*Monster-game.py*

import pygame

from goblin import Goblin

from hero import Hero

from monster import Monster

# Key Stroke Variables

KEY\_UP = 5

KEY\_DOWN = 6

KEY\_LEFT = 7

KEY\_RIGHT = 8

KEY\_ENTER = 9

# Other Variables

level\_count = 1

bg\_sound\_path = "./sounds/music.wav"

lose\_sound\_path = "./sounds/lose.wav"

win\_sound\_path = "./sounds/win.wav"

bg\_image\_path = "./images/background.png"

window\_width = 512

window\_height = 480

def restart\_level(monster, hero, goblin1, goblin2, goblin3):

monster.monster\_restart()

hero.hero\_restart()

goblin1.goblin\_restart()

goblin2.goblin\_restart()

goblin3.goblin\_restart()

def main():

# Initialize pygame

pygame.init()

# Importing image

background\_image = pygame.image.load(bg\_image\_path)

# Importing and Intialize Sound

pygame.mixer.init()

bg\_sound = pygame.mixer.Sound(bg\_sound\_path)

lose\_sound = pygame.mixer.Sound(lose\_sound\_path)

win\_sound = pygame.mixer.Sound(win\_sound\_path)

# Set up the screen (drawing surface for the game)

screen = pygame.display.set\_mode((window\_width, window\_height))

# Game Window Title

pygame.display.set\_caption('Monster Game')

# Calls the Clock object and sets it to variable clock (for shorthand purposes)

clock = pygame.time.Clock()

#################### Game initialization ####################

# Created a monster instance

game\_monster = Monster(120, 120, 3, 3)

game\_hero = Hero(256, 240)

game\_goblin1 = Goblin(360, 360)

game\_goblin2 = Goblin(360, 360)

game\_goblin3 = Goblin(360, 360)

# Sprite Groups

monster\_group = pygame.sprite.Group()

monster\_group.add(game\_monster)

hero\_group = pygame.sprite.Group()

hero\_group.add(game\_hero)

goblin\_group = pygame.sprite.Group()

goblin\_group.add(game\_goblin1)

goblin\_group.add(game\_goblin2)

goblin\_group.add(game\_goblin3)

# FPS settings

FPS = 60

# Music Start

bg\_sound.play()

#################### While loop used to have the game continuously run ####################

stop\_game = False

while not stop\_game:

# Event is for user input such as keypress to clicks

for event in pygame.event.get():

# Key Strokes Movement (Hero Movement and Enter Key)

if event.type == pygame.KEYDOWN:

if event.key == KEY\_DOWN:

if game\_hero.dead == False:

game\_hero.dir\_y = 3

elif event.key == KEY\_UP:

if game\_hero.dead == False:

game\_hero.dir\_y = -3

elif event.key == KEY\_LEFT:

if game\_hero.dead == False:

game\_hero.dir\_x = -3

elif event.key == KEY\_RIGHT:

if game\_hero.dead == False:

game\_hero.dir\_x = 3

elif event.key == KEY\_ENTER:

print("Enter Works")

restart\_level(game\_monster, game\_hero, game\_goblin1, game\_goblin2, game\_goblin3)

monster\_group.add(game\_monster)

goblin\_group.add(game\_goblin1)

goblin\_group.add(game\_goblin2)

goblin\_group.add(game\_goblin3)

bg\_sound.play()

if event.type == pygame.KEYUP:

if event.key == KEY\_DOWN:

game\_hero.dir\_y = 0

elif event.key == KEY\_UP:

game\_hero.dir\_y = 0

elif event.key == KEY\_LEFT:

game\_hero.dir\_x = 0

elif event.key == KEY\_RIGHT:

game\_hero.dir\_x = 0

# Event that if user clicks exit

if event.type == pygame.QUIT:

stop\_game = True

# Game logic

# Renders background, monster, and hero image

screen.blit(background\_image, [0, 0])

screen.blit(game\_hero.image, [game\_hero.x, game\_hero.y])

screen.blit(game\_monster.image, [game\_monster.x, game\_monster.y])

screen.blit(game\_goblin1.image, [game\_goblin1.x, game\_goblin1.y])

screen.blit(game\_goblin2.image, [game\_goblin2.x, game\_goblin2.y])

screen.blit(game\_goblin3.image, [game\_goblin3.x, game\_goblin3.y])

# Hero Movement

game\_hero.character\_movement()

game\_hero.character\_fence(1)

game\_hero.rect\_update()

# Monster Movement

game\_monster.monster\_random\_movement()

game\_monster.character\_movement()

game\_monster.character\_fence(2)

game\_monster.rect\_update()

# Goblin Movement

game\_goblin1.monster\_random\_movement()

game\_goblin1.character\_movement()

game\_goblin1.character\_fence(2)

game\_goblin1.rect\_update()

game\_goblin2.monster\_random\_movement()

game\_goblin2.character\_movement()

game\_goblin2.character\_fence(2)

game\_goblin2.rect\_update()

game\_goblin3.monster\_random\_movement()

game\_goblin3.character\_movement()

game\_goblin3.character\_fence(2)

game\_goblin3.rect\_update()

# Sprite Collide

collide\_monster = pygame.sprite.spritecollide(game\_hero, monster\_group, True)

collide\_goblin = pygame.sprite.spritecollide(game\_hero, goblin\_group, True)

if collide\_monster:

bg\_sound.stop()

game\_hero.x = 230

game\_hero.y = 200

game\_monster.is\_dead()

win\_sound.play()

if collide\_goblin:

bg\_sound.stop()

game\_monster.x = 500

game\_monster.y = 500

game\_hero.hero\_dead()

lose\_sound.play()

game\_monster.dir\_x = 0

game\_monster.dir\_y = 0

if game\_monster.dead == True:

font = pygame.font.Font(None, 20)

play\_again\_text = font.render("Press `Enter` to Play Again!", True, (0, 0, 0))

screen.blit(play\_again\_text, [172, 220])

game\_hero.x = 236

game\_hero.y = 185

if game\_hero.dead == True:

font = pygame.font.Font(None, 20)

play\_again\_text = font.render("Press `Enter` to Play Again!", True, (0, 0, 0))

screen.blit(play\_again\_text, [172, 220])

game\_goblin1.x = 236

game\_goblin1.y = 185

game\_goblin2.x = 236

game\_goblin2.y = 185

game\_goblin3.x = 236

game\_goblin3.y = 185

game\_hero.dir\_x = 0

game\_hero.dir\_y = 0

# Game display

pygame.display.update()

# Set the tick rate to 60 ms, which means the game runs at 60 FPS

clock.tick(FPS)

pygame.quit()

if \_\_name\_\_ == '\_\_main\_\_':

main()

*Goblin.py*

import random

import pygame

from monster import Monster

class Goblin(Monster):

goblin\_image = pygame.image.load("./images/goblin.png")

def \_\_init\_\_(self, x, y):

Monster.\_\_init\_\_(self, x, y, 1, 1)

self.image = pygame.image.load("./images/goblin.png").convert\_alpha()

self.rect = self.image.get\_rect()

self.rect.center = self.x, self.y

def goblin\_restart(self):

self.image = pygame.image.load("./images/goblin.png").convert\_alpha()

self.x = random.randint(40, 440)

self.y = random.randint(40, 400)

self.dir\_x = 1

self.dir\_y = 1

self.dead = False