INTRODUCTION TO COMPUTATIONAL PHYSICS

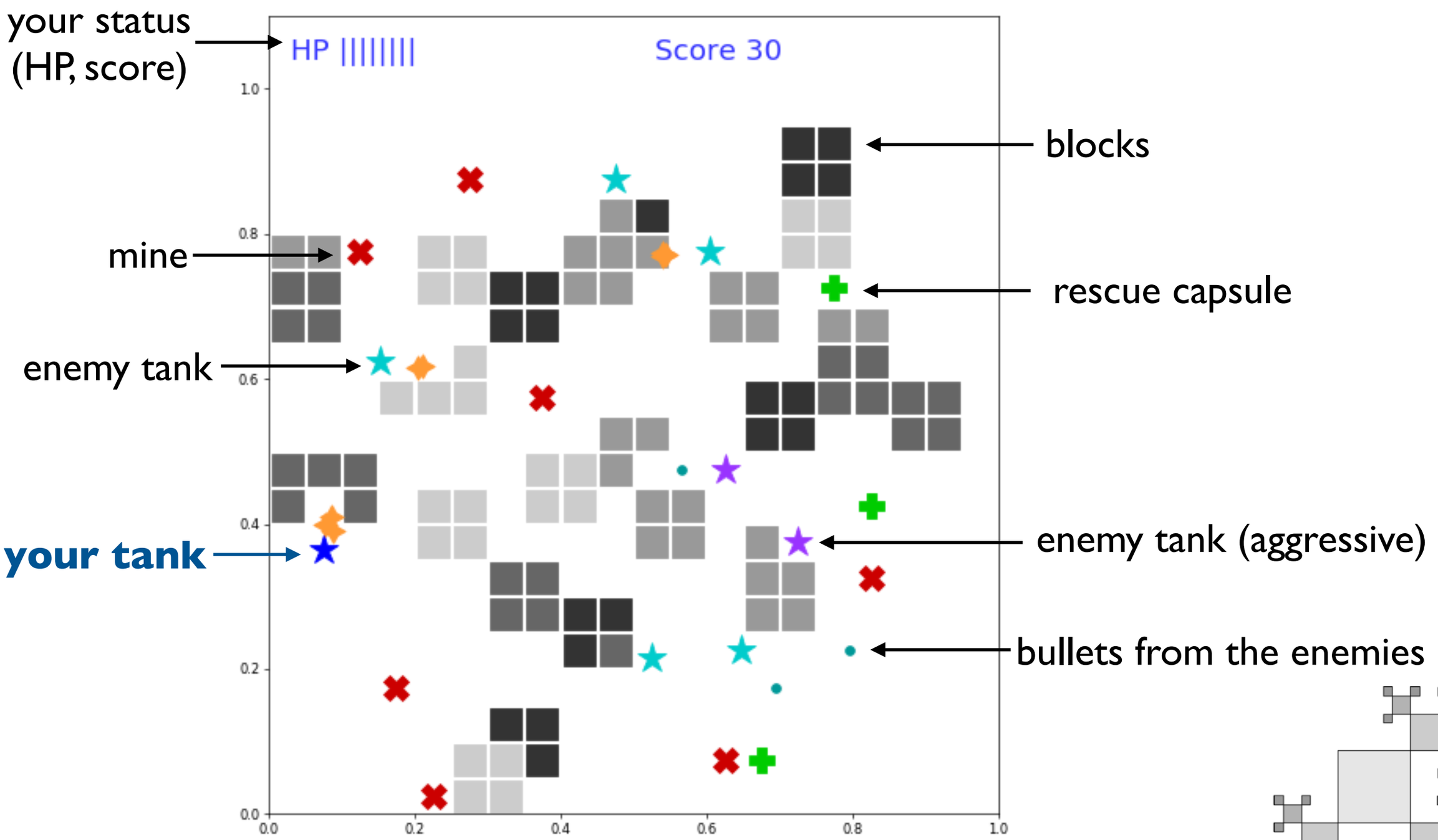
Kai-Feng Chen
National Taiwan University

WHAT ARE WE GOING TO DO?

- We will have a **Video Game Competition**!
- And we are going to play a not-so-classical **"battle city"-like game**.
- All you need to do is derive a good AI program to control your tank, hide from the attacks, and shoot your enemy down!
- We are going to run this competition, and who gets the **higher score** will win!

WELL, WE DO NOT HAVE "REAL" GRAPHICS...

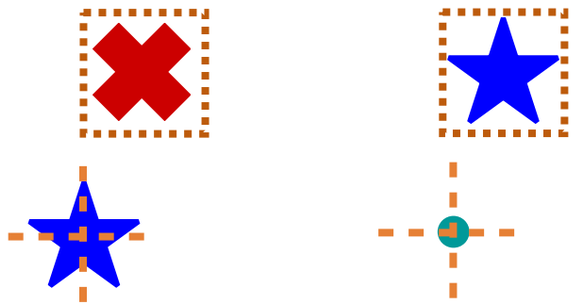
- Once you execute the `game01.py/game01.cc` code:



COORDINATION SYSTEM

- All of the tanks are limited in the area of (0,0) - (1,1).
- The basic grid size is 0.05 — ie. as your tank moves, it would move in the step of 0.05 (and it takes 10 frames!).
- All of the objects are in the dimension of 0.05×0.05, too (while the bullet has no size!)

- Collisions of bullets (or mines / rescue caps) with your tank or the enemy tanks are identified whenever the **center** of objects enter the **square region**.



TYPES OF OBJECTS

- Code 0/1 - your/enemy's cannon-shot, speed = 0.01/frame

Code 2 - player tank, speed = 0.05 / 10 frames

Code 3 - enemy tank, speed = 0.05 / 20 frames, HP = 2;
score = 1000 points + 10 points / hit

Code 4 - stronger enemy tank, speed = 0.05 / 10 frames, HP = 3;
score = 2000 points + 10 points / hit

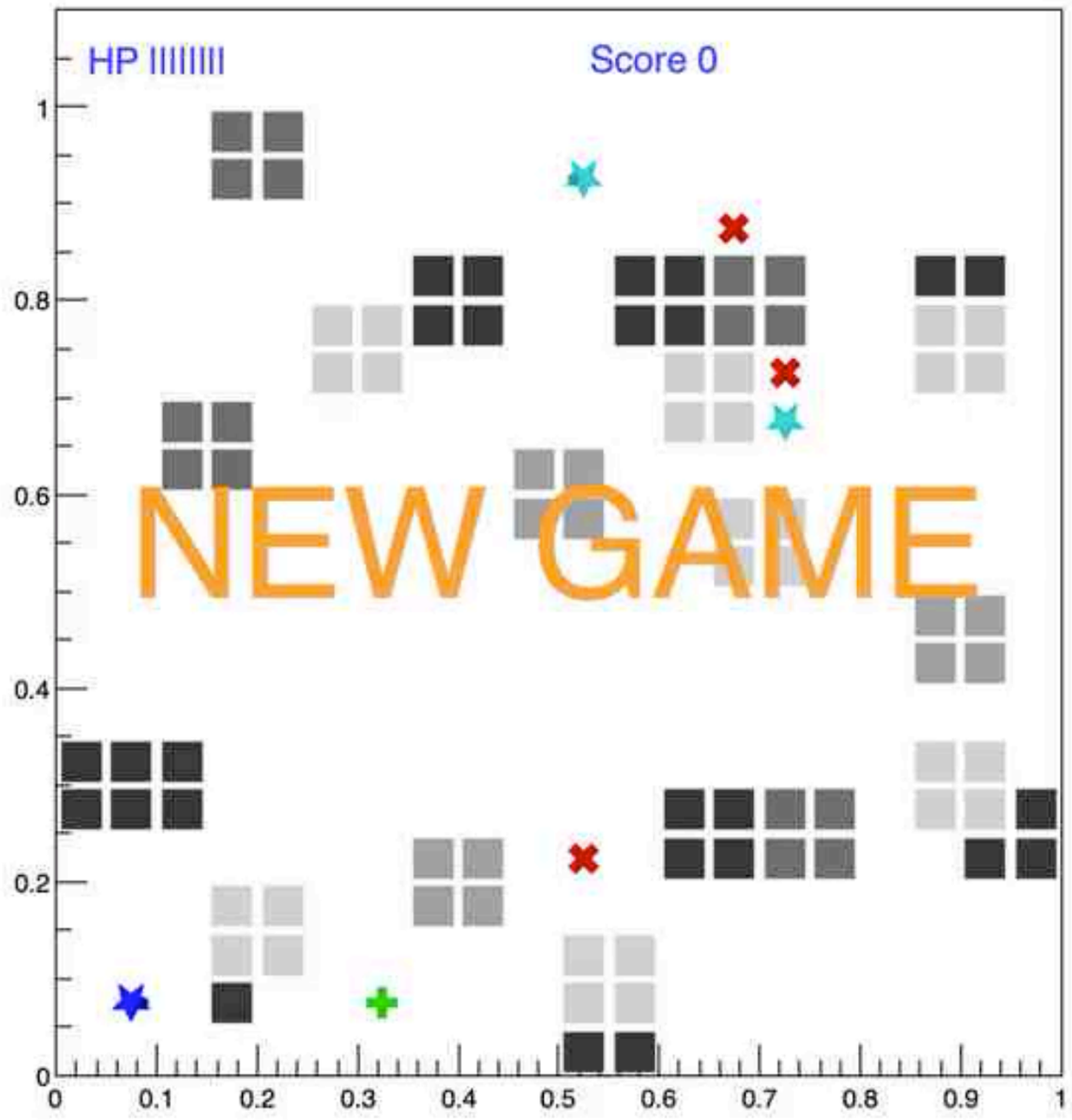
Code 5 - block, HP = 4, 3, 2, 1;
score =10 points / hit

Code 6 - mine;
score =50 points

Code 7 - rescue capsule;
score =100 points

One hit (*by a cannon-shot or a mine*) = **HP – 1**
Acquiring a rescue capsule = **HP + 2**
Your initial HP is **8**, maximum **16**.

DEMO PLAYBACK



- This is how the program *"play the game"* for you.
- Remarks:
  - Your tank starts from left-bottom corner.
  - Once you destroy all of the enemy tanks, the map will be reseted and level up.
  - Surely…when level up the enemies will become more *"aggressive"*!

PLAYER TEMPLATE

The **decision function** will be called by the main program **to decide the action** and you need to return an integer **"action command"**.

partial **player_module.py**

```
class player_module:

    # Constructor, allocate any private date here
    def __init__(self):
        self.player_x, self.player_y = 0., 0.

    # Please update the banner according to your information
    def banner(self):
        print('----------------------------')
        print('Author: your_name_here')
        print('ID: bxxxxxxxx')
        print('----------------------------')

#   Decision making function for moving your tank
    def decision(self, score, player_hp, player_status, code, x, y, dx, dy):
```

← Constructor

← Put your name and ID here

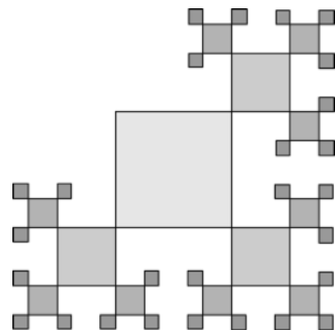
↙ the main decision function

Action Commands

0 - Idle

1 - move down (y–)
2 - move up (y+)
3 - move left (x–)
4 - move right (x+)

5 - fire down (y–)
6 - fire up (y+)
7 - fire left (x–)
8 - fire right (x+)



PLAYER TEMPLATE (II)

- Surely there is a C++ version of the player template, too!

partial **player_module.h**

```
class player_module {
public:
    double player_x, player_y;

    // Constructor, allocate any private date here
    player_module() : player_x(0.), player_y(0.) {}

    // Please update the banner according to your information
    void banner() {
        printf("------------------------\n");
        printf("Author: your_name_here\n");
        printf("ID: bxxxxxxxx\n");
        printf("------------------------\n");
    }

/*  Decision making function for moving your tank, toward next decision
frame: */
    int decision(int score, int player_hp, int player_status,
                 std::vector<int> &code,
                 std::vector<double> &x, std::vector<double> &y,
                 std::vector<double> &dx, std::vector<double> &dy) {
```

← Constructor

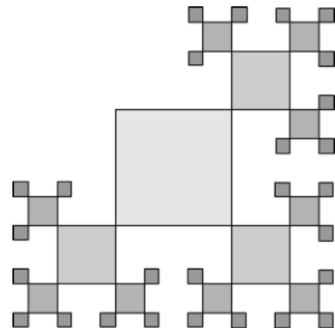
← Put your name and ID here

↙ the main decision function

INPUT ARGUMENTS

The **player_status** indicates whether your tank can move / fire, if you return an invalid action command, it will simply do nothing (idle = 0).

- **score** (*integer*): current score
- **player_hp** (*integer*): HP of your tank
- **player_status** (*integer*)
  - 1 - can only move
    Hence you should only return 1-4 (moving toward 4 directions)
  - 2 - can move / can fire cannon
    Hence you can return 1-4 (moving) or 5-8 (cannon-fire)
- **code** (*list/vector of integer*): type of objects
  (see the definitions given in the earlier slide, e.g. 0/1 are cannon-shots, 2 is player tank, etc.)
- **x/y/dx/dy** (*list/vector of doubles*) =
  coordinate of the objects, and current displacement



HAVE FUN!

- We will have two rounds of competitions:
  - **First Round**: we will run your code and calculate the average scores from multiple trials.
    - ➡ If your average scores beat half of the participants, you will win a trophy!
    - ➡ We will invite the **top 4 players** to enter the final round (*you can provide an updated code if you wish*).
  - **Final Round**: we will have a direct "play" in the class and see **who gets the highest score!**
- Please provide your code before **Dec/23** for the first round; the final round show will be held on **Dec/30** (*with your final code*).