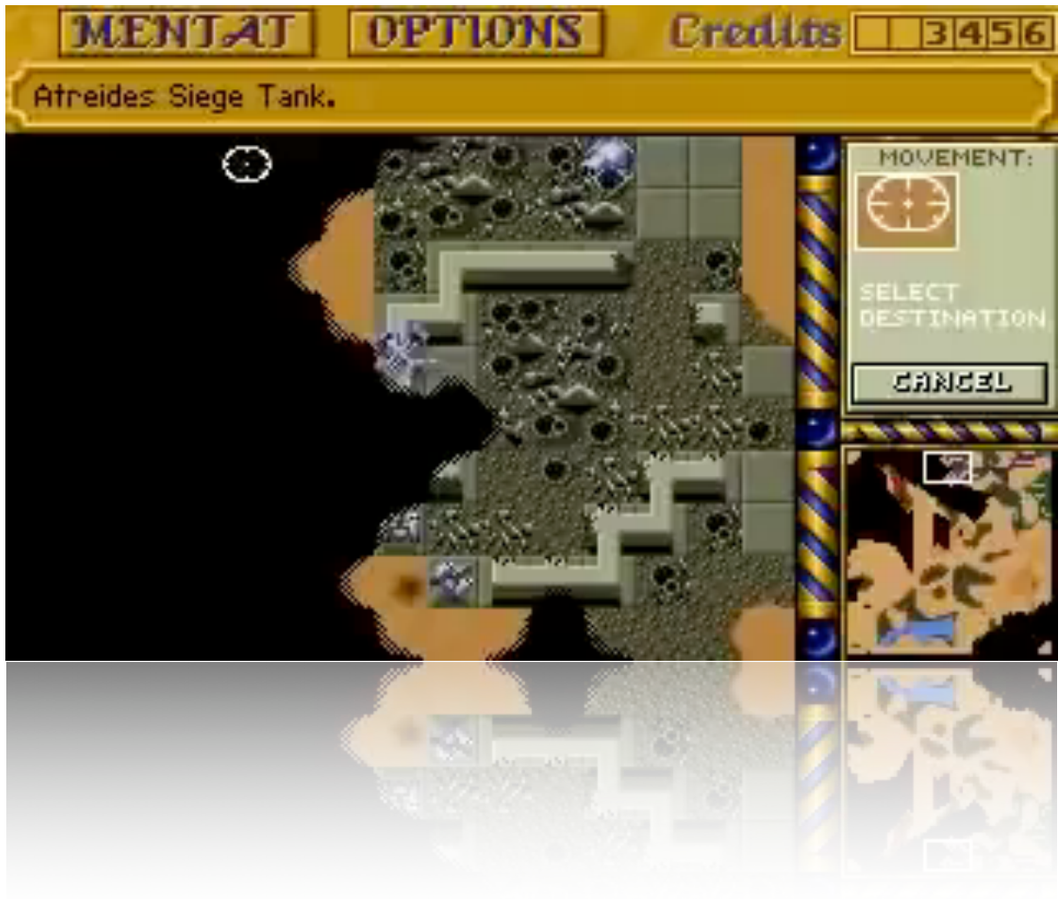
INTRODUCTION TO COMPUTATIONAL PHYSICS

Kai-Feng Chen
National Taiwan University

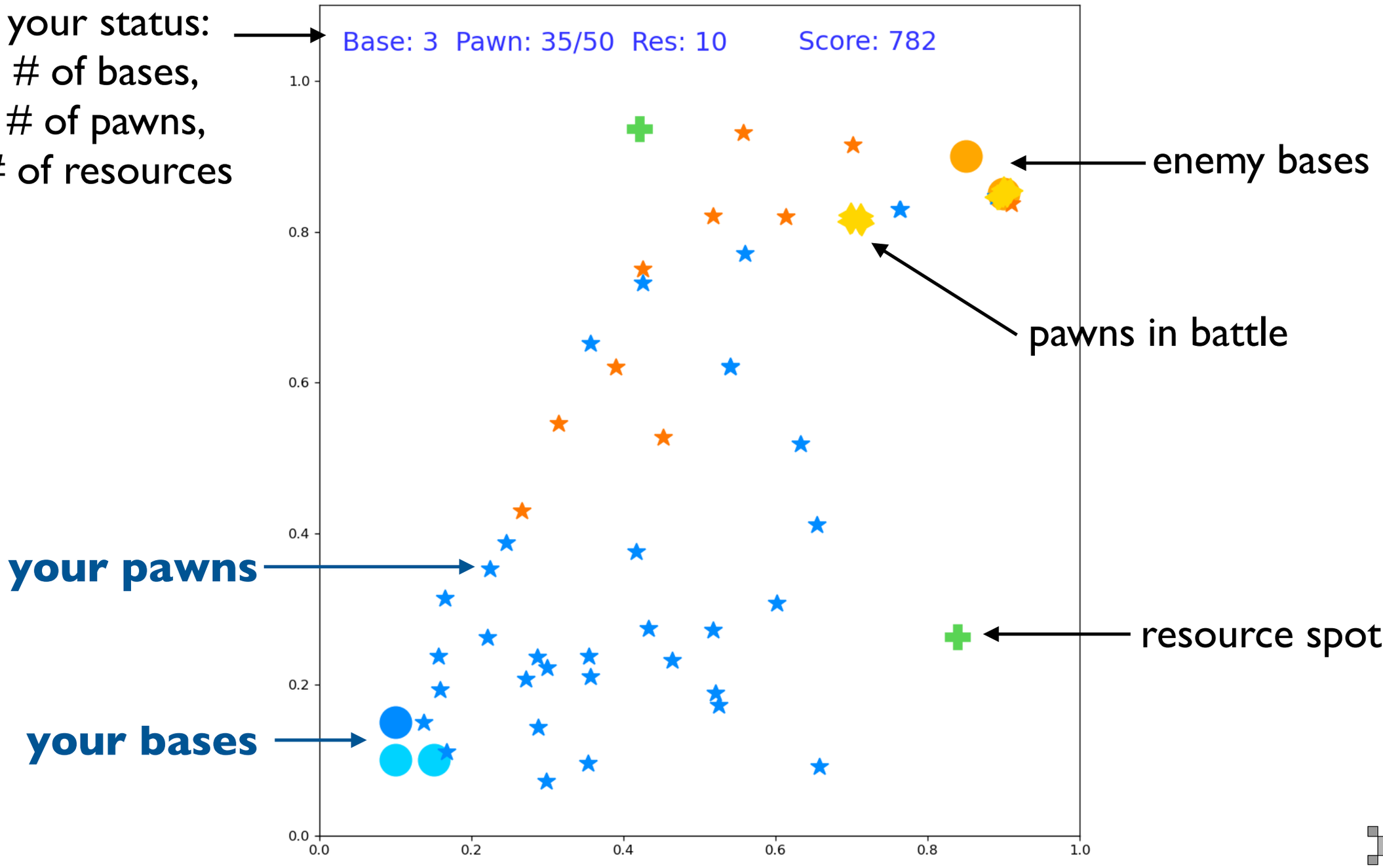
WHAT ARE WE GOING TO DO?

- We will have a **Video Game Competition**, again!
- We are going to play a simple **real-time-strategy**-like game!
- All you need to do is derive a good AI program to control your "pawn", collect resources, protect/construct your "bases"!
- 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 `game02.py/game02.cc` code:

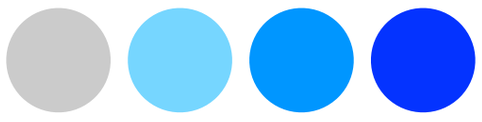


TYPES OF OBJECTS

Note the pawn moving speed is 0.005 (0.001) per frame normally (in attacking).

Code 0 - player pawn (*free*), radius = 0.01 / hp = 10

Code 1 - player pawn (*busy*), radius = 0.01 / hp = 10



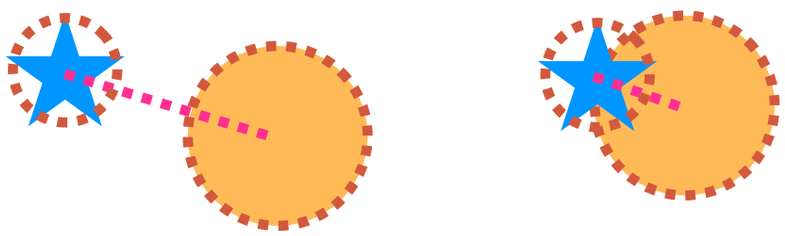
Code 2 - player base (*lv0/lv1/lv2/lv3*), radius = 0.02, hp = 50-250

Code 3 - enemy pawn, radius = 0.01 / hp = 10

Code 4 - enemy base (*lv0/lv1/lv2/lv3*), radius = 0.02, hp = 50-250

Code 5 - resource spot, radius = 0.01, hp = 10-20

- All of the objects are considered as a circle and stay within (0,0) - (1,1).
- Contacts of pawns (bases or resources spots) is decided by the distance between two objects, ie. if the distance is less than the larger radius of the two.



RULES AND REGULATIONS

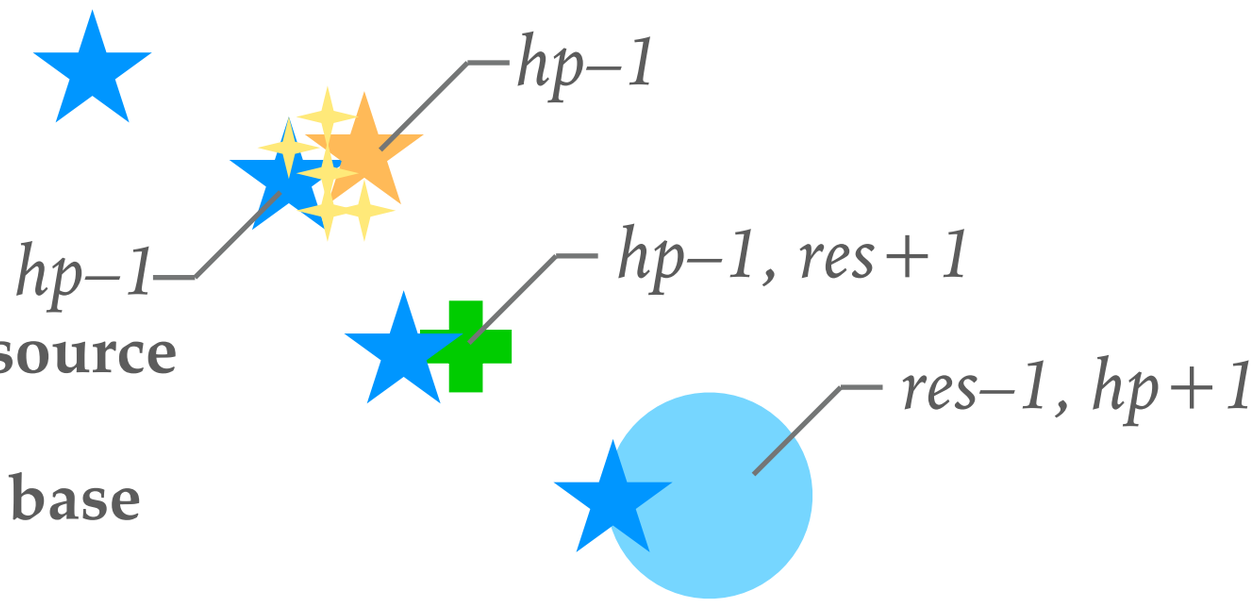
- You can assign a command to your “free” pawns.
- The full command is consistent with a **target location (x, y)** and an **action**:

0 - standby

1 - attack

2 - collect resource

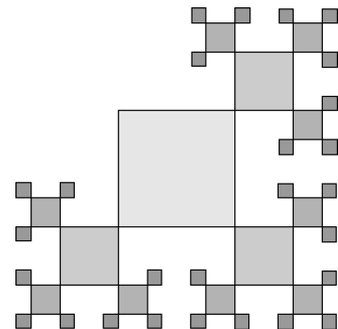
3 - construct base



- However if your pawn get contacted with enemy pawn or base, it will automatically switched to “attack” command.
- When pawns are attacking, any contacted enemy pawn or base will have **50% chance to reduce 1 HP per frame**. When an enemy pawn (base) is destroyed, you won 20 (200) score points.
- When pawns are collecting resource, **every frame can obtain 1 resource point and reduce the HP of the resource point by 1**.

RULES AND REGULATIONS (II)

- You can upgrade your existing base or construct a new one at any place.
- When pawns are construct the base, every frame can **add 1 HP to the base and reduce the resource points by 1**.
- The base level depends on the HP of the base:
  - **LV0** (*HP<50*) - inactive, cannot host any pawn
  - **LV1** (*50≤ HP<150*) - every 40 frames can generate 1 pawn, can host 15 pawns at most.
  - **LV2** (*150≤ HP<200*) - every 32 frames can generate 1 pawn, can host 20 pawns at most.
  - **LV3** (*200≤ HP≤250*) - every 24 frames can generate 1 pawn, can host 25 pawns at most.
- More (high level) bases = more pawns = more powerful!



DEMO PLAYBACK

- This is how the program "play the game" for you.
- Remarks:
  - Your base/pawns starts from left-bottom corner.
  - Once you *kills all of the enemies* (or *reaches some certain amount of scores*), the **game level upgrades** and new enemy bases/pawns will be regenerated randomly again.
  - So does the resource spots!

PLAYER TEMPLATE

The **decision function** will be called by the main program every frame to **decide the actions** of your free pawns.

partial player_module.py

```
class player_module:

    # Constructor, allocate any private data here
    def __init__(self):
        pass

    # 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 the action of your pawns
def decision(self, score, resource_player, resource_enemy,
             code, hp, x, y, target_cmd, target_x, target_y):
```

← Constructor

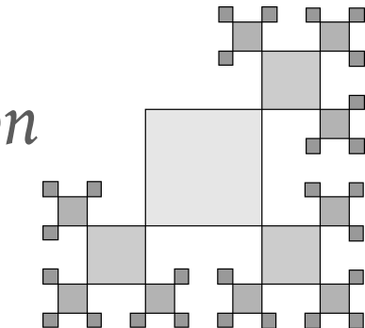
← Put your name and ID here

↙ the main decision function

Returning command data

target_cmd: *action command number = 0,1,2,3*

target_x, target_y: *move your pawn to the target location*



PLAYER TEMPLATE (II)

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

partial **player_module.h**

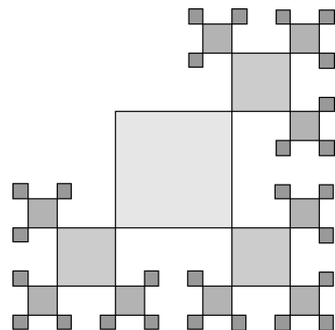
```
class player_module {
public:
  // Constructor, allocate any private data here
  player_module() {}  ← Constructor

  // Please update the banner according to your information
  void banner() {
      printf("--------------------------\n");
      printf("Author: your_name_here\n");
      printf("ID: bxxxxxxxx\n");
      printf("--------------------------\n");
  }
                                                  ↙ the main decision function
  // Decision making function for the action of your pawns
  void decision(int score, int resource_player, int resource_enemy,
      std::vector<int> &code,
      std::vector<int> &hp,
      std::vector<double> &x, std::vector<double> &y,
      std::vector<int> &target_cmd,
      std::vector<double> &target_x, std::vector<double> &target_y) {
```

← Put your name and ID here

INPUT ARGUMENTS

Your decision function should change the contents of **target_cmd** & **target_x/y**, for all of those code=0 (*free player pawns*).

- **score** (*integer*): current score
- **resource_player** (*integer*): your resource points
- **resource_enemy** (*integer*): enemy's resource points
- **code** (*list/vector of integer*): type of objects
  (see the definitions given in the earlier slide, e.g. 0/1 are player's pawn, 2 is player base, etc.; note the list/vector always starts from code=0 objects)
- **hp** (*list/vector of integer*): HP of the objects
- **x/y** (*list/vector of doubles*): coordinate of the objects
- **target_cmd** (*list/vector of integer*): target action commands (for code=0 objects only)
- **target_x/target_y** (*list/vector of doubles*): target location (for code=0 objects only)



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** (*note, this trophy is the same one for another ML competition, even if you win both it only counts once*)!
    - ➡ 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 **June/3** for the first round; the final round show will be held on **June/10** (*with your final code*).