Assign 6

Object

Requirements

★ 完成 Cabbage Class：繼承 Item Class、沿用其建構式並利用 w, h, x, y 變數覆寫 display() 與 checkCollision(Player player) 方法。蔬菜運作方式參考先前作業要求。土撥鼠碰到蔬菜後，將 boolean isAlive 設為 false；如果 isAlive 為 false 則不顯示圖片亦不偵測碰撞。

★ 完成 Clock Class：繼承 Item Class、沿用其建構式並利用 w, h, x, y 變數覆寫 display() 與 checkCollision(Player player) 方法。時鐘運作方式參考先前作業要求。土撥鼠碰到時鐘後，將 boolean isAlive 設為 false；如果 isAlive 為 false 則不顯示圖片亦不偵測碰撞。

★ 利用一個 Item[] items 陣列儲存蔬菜與時鐘，陣列長度為 6，蔬菜與時鐘生成方式改為每四層只隨機出現兩者之一。在遊戲進行過程中，利用 items 陣列呼叫各個道具的 display() 與 checkCollision(Player player) 方法。

Requirements

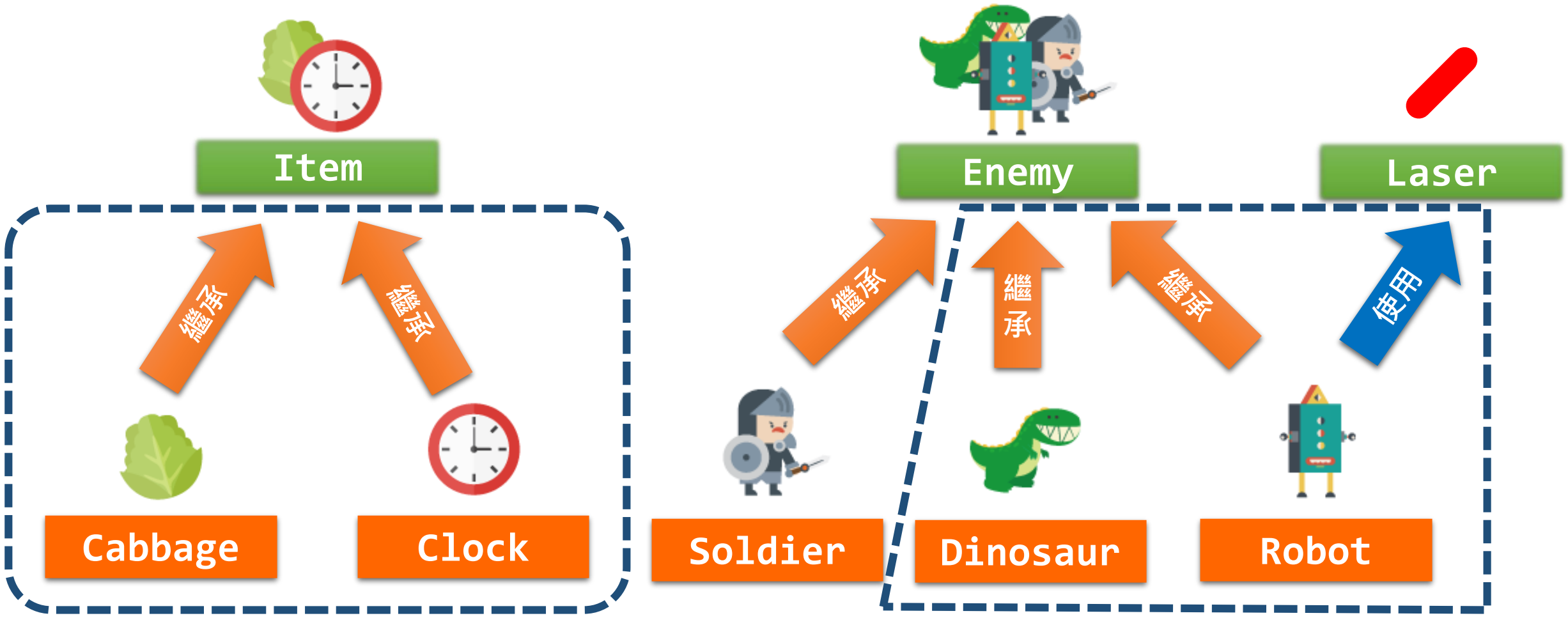
★★ 完成 Dinosaur Class：繼承 Enemy Class、沿用其建構式並覆寫 display() 與 update() 方法。恐龍出現在第 9 - 16 層，位置隨機條件與士兵相同，並在 Enemy[] enemies 陣列儲存恐龍物件。當恐龍碰到畫面邊界時會往相反方向移動，並改變圖片顯示方向；正常速度是士兵的一半，但在偵測到土撥鼠在同一層且在前方時速度則乘五倍，而當土撥鼠離開偵測範圍後恢復原本速度。

★★ 完成 Robot Class：繼承 Enemy Class、沿用其建構式並覆寫 display() 與 update() 方法。機器人出現在第 17 - 24 層，位置隨機條件與士兵相同，並在 Enemy[] enemies 陣列儲存機器人物件。當機器人碰到畫面邊界時會往相反方向移動，並改變圖片顯示方向；速度與士兵相同，偵測到土撥鼠在上下兩層內 (-2 ~ +2) 且在前方時則暫停移動，而當土撥鼠離開偵測範圍後恢復移動。

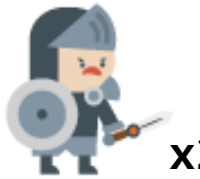
★★★在 Robot 中使用 Laser 物件，當機器人偵測到土撥鼠時（如前述）從手中朝玩家中心處發射一道雷射光。雷射光每隔三秒發射一次。

各類別繼承關係

藍框內容為同學需要完成的部分



主程式邏輯



x2

total: 6

x2

x2

x?

total: 6

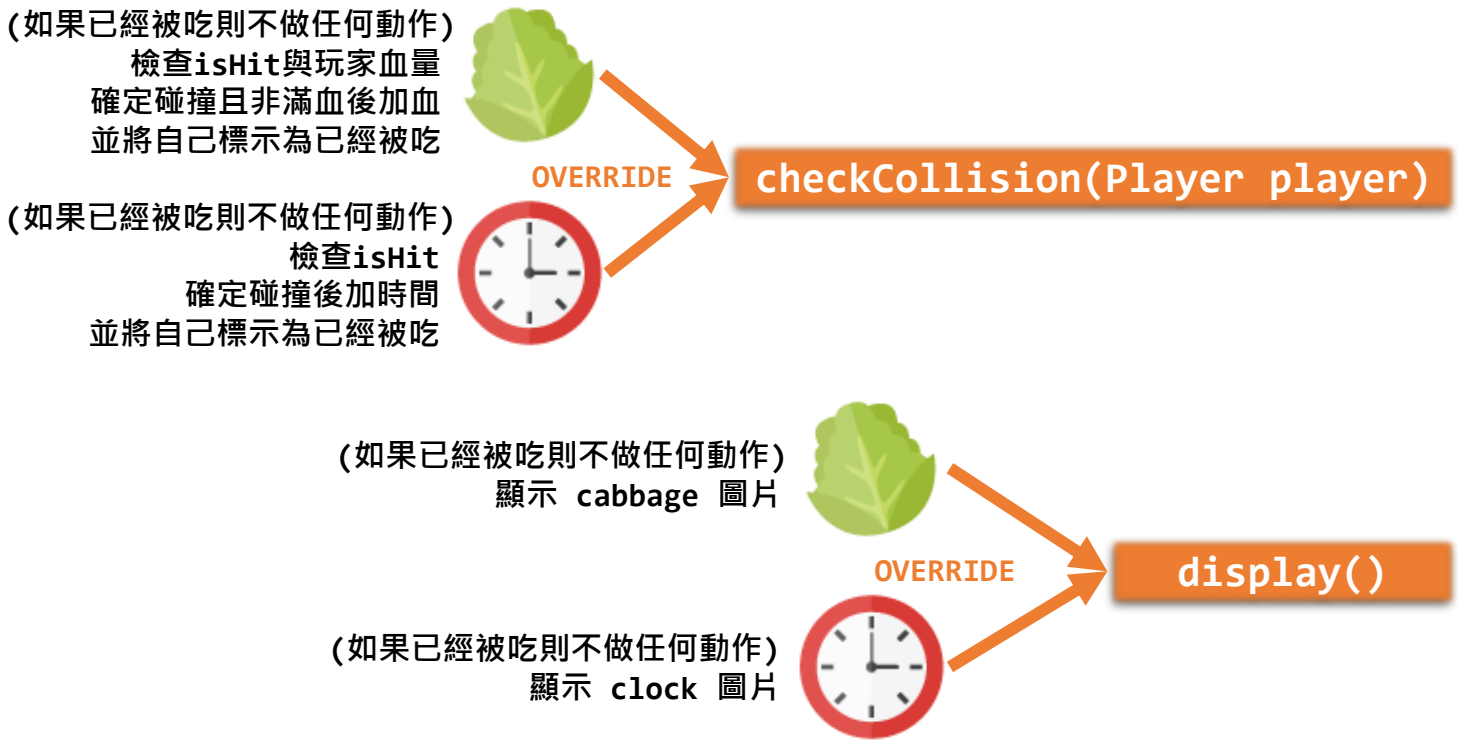
x?

Enemy[] enemies

Item[] items

```
void setup(){
  . . . . . .
  initGame();
}

void initGame(){
  . . . . . .

  將 Enemy[] enemies 陣列初始化
  (Soldier部分已完成)

  將 Item[] items 陣列初始化

}
```

程式邏輯




```
void draw(){
  . . . . . .

  (line 238)
  對 items 裡面每個 Item
    呼叫檢查玩家碰撞的方法
    呼叫顯示的方法

  (line 245)
      已完成，不須改動
      只需在initGame()正確地將恐龍與機器人放入 enemies
  對 enemies 裡面每個 Enemy
    呼叫更新位置與狀態的方法
    呼叫顯示圖片的方法
    呼叫檢查玩家碰撞的方法

  . . . . . .
}
```

程式邏輯

持續往右移動
超出右邊時回到左邊

朝面對方向持續移動
撞到邊界後往反方向移動
偵測到土撥鼠則加速移動

OVERRIDE

朝面對方向持續移動
撞到邊界後往反方向移動
偵測到土撥鼠則停止並試圖發射雷射光
更新雷射光冷卻時間
呼叫雷射光更新位置

顯示 soldier 圖片

顯示 dinosaur 圖片並視面對方向做反轉

OVERRIDE

顯示 robot 圖片並視面對方向做反轉
呼叫雷射光顯示

除了自己檢查以外
也呼叫雷射光檢查

OVERRIDE

update()

display()

checkCollision(Player player)

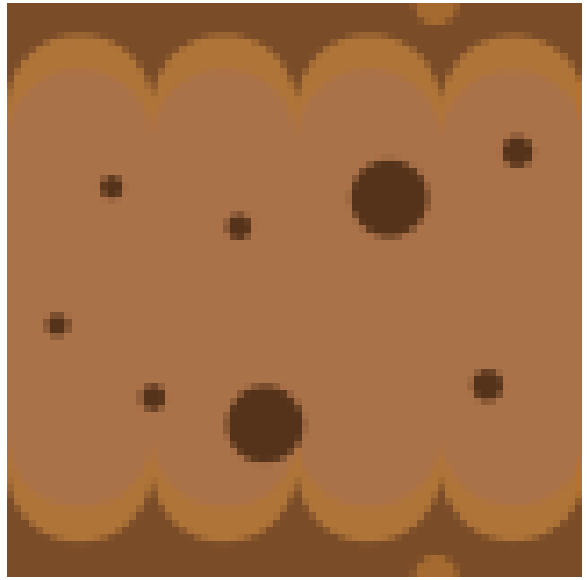
其他兩種敵人碰撞檢查與反應都一樣
不須另外覆寫

```
void draw(){
  . . . . . .

  (line 238)
  對 items 裡面每個 Item
    呼叫檢查玩家碰撞的方法
    呼叫顯示的方法

  (line 245)
    已完成，不須改動
    只需在initGame()正確地將恐龍與機器人放入 enemies
  對 enemies 裡面每個 Enemy
    呼叫更新位置與狀態的方法
    呼叫顯示圖片的方法
    呼叫檢查玩家碰撞的方法

  . . . . . .
}
```

Player Class (已包含於程式碼)

```
float x, y;
float w = SOIL_SIZE, h = SOIL_SIZE;
int col, row; // 判斷機器人攻擊範圍時可使用
int health = 2;

void update(){
    // 內含移動、挖土、顯示等程式碼
}

void hurt(){
    // 扣血後放回初始位置並填回下方土壤
    // 如血量歸零則結束遊戲
}

Player(){
    // 初始化位置與血量
}
```

Soil Class (已包含於程式碼)



```
int col, row;
int health;

void display(){
    // 依血量與深度顯示圖片
}

Soil(int col, int row, int health){
    // 初始化土壤
}
```

Item Class (已包含於程式碼)

```
boolean isAlive;
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;

void display(){}
void checkCollision(Player player){}

Item(float x, float y){
    isAlive = true;
    this.x = x;
    this.y = y;
}
```

Cabbage Class

繼承 Item 的變數（不須重新宣告）

```
boolean isAlive;
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;
```

如 isAlive = false 則不須執行

display()：顯示蔬菜圖片

checkCollision(Player player)：

將自己與 player 的 xywh 代入 isHit()

如果偵測到碰撞且 player.health 未達上限

則加血並將自己的 isAlive 設為 false

宣告一個 Cabbage 建構式來沿用 Item 建構式

Clock Class

繼承 Item 的變數（不須重新宣告）

```
boolean isAlive;
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;
```

如 isAlive = false 則不須執行

display()：顯示時鐘圖片

checkCollision(Player player)：

- 將自己與 player 的 xywh 代入 isHit()
- 如果偵測到碰撞則呼叫 addTime(float seconds)
- 並將自己的 isAlive 設為 false

宣告一個 Clock 建構式來沿用 Item 建構式

Enemy Class (已包含於程式碼)

```
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;

void checkCollision(Player player){
    // 如果偵測到碰撞則呼叫player.hurt()
}

void display(){}
void update(){}

Enemy(float x, float y){
    this.x = x;
    this.y = y;
}
```

Soldier Class (已包含於程式碼)

繼承 Enemy 的變數（不須重新宣告）

```
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;
```

```
float speed = 2f;
```

display()：顯示士兵圖片
update()：
    移動士兵，在超出畫面右邊時放回左邊

沿用 Enemy 建構式

Dinosaur Class

繼承 Enemy 的變數（不須重新宣告）

```
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;
```

```
// 發現玩家時速度倍率
final float TRIGGERED_SPEED_MULTIPLIER = 5;
```

display()：依目前移動方向顯示恐龍圖片

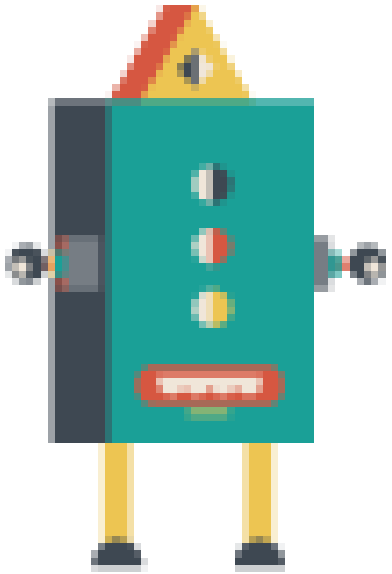
update()：

移動恐龍，碰到畫面邊界時會往相反方向移動

當面對土撥鼠、發現土撥鼠與自己在同一層則加速

沿用 Enemy 建構式

Robot Class

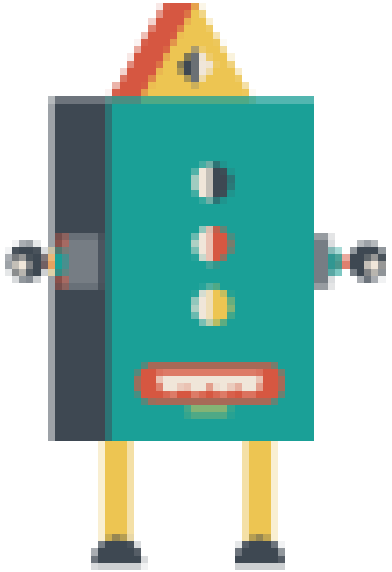


繼承 Enemy 的變數（不須重新宣告）

```
float x, y;
float w = SOIL_SIZE;
float h = SOIL_SIZE;
```

```
// 偵測土撥鼠的上下範圍，2 代表 -2 ~ +2
final int PLAYER_DETECT_RANGE_ROW = 2;

// 雷射的冷卻時間，需要再自行宣告一個timer來計時
final int LASER_COOLDOWN = 180;

// 需要自行宣告一個 Laser 變數

// 雷射光發射位置與圖片左上角的 Y 偏移量
final int HAND_OFFSET_Y = 37;

// 往前（右）時
// 雷射光發射位置與圖片左上角的 X 偏移量
final int HAND_OFFSET_X_FORWARD = 64;

// 往後（左）時
// 雷射光發射位置與圖片左上角的 X 偏移量
final int HAND_OFFSET_X_BACKWARD = 16;
```

Robot Class



display()：

依目前移動方向顯示機器人圖片、呼叫雷射光顯示

update()：

移動機器人，碰到畫面邊界時會往相反方向移動

當面對土撥鼠、發現土撥鼠在上下偵測範圍內

X：以發射處為參考點與玩家中心點做比較（需考慮面對方向）

Y：直接使用 player 的 row 變數判斷

則停止移動並嘗試朝土撥鼠中心點發射雷射光

如果雷射仍在冷卻則停留在原地不動

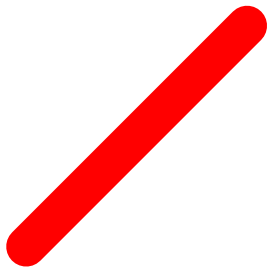
最後呼叫雷射光更新位置

checkCollision(Player player)：

除了自己檢查碰撞外也呼叫雷射光檢查

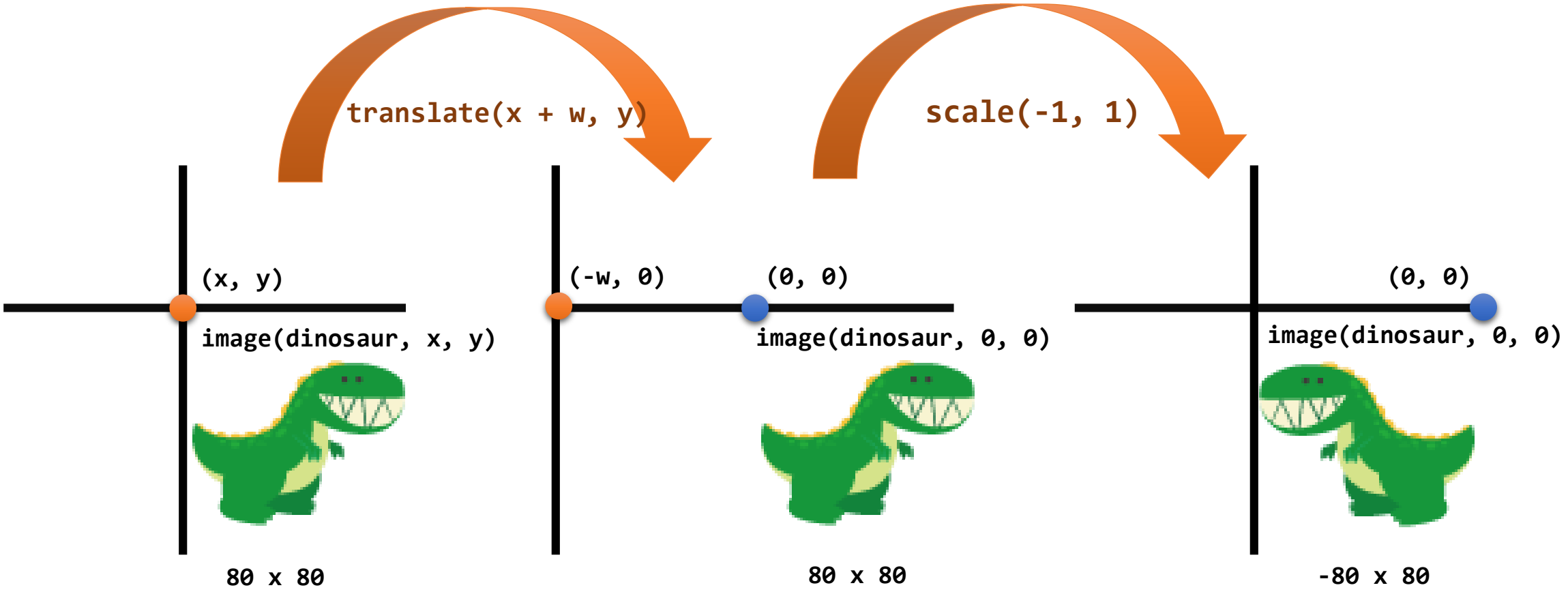
沿用 Enemy 建構式後呼叫 Laser 建構式

Laser Class (已包含於程式碼)



```
boolean isAlive;
float x, y;
float originX, originY;
float angle;
static final float maxLength = 20f;
float speed = 4f;

// 依角度來移動雷射光
void update()

// 顯示雷射光
void display()

// 判斷雷射光的前端是否碰到土撥鼠
void checkCollision(Player player)

// 發射雷射光，需要告訴雷射光從哪裡往哪裡發射才能計算角度並正確顯示發射動畫
void fire(float originX, float originY, float targetX, float targetY)

Laser(){
    // 預設狀態為 false，所以在使用建構式後雷射不會自動發射或顯示
    // 呼叫 fire() 後會切為 true
    isAlive = false;
}
```

如何左右反轉圖片



記得使用pushMatrix()與popMatrix()避免影響其他圖片顯示
可參考水族箱課堂練習程式碼

Demo 影片

https://www.youtube.com/watch?v=zR4KohfjLXw