A-1 encapsulation 封裝 -> private 用 set/get 變動 取值 避免其他地方直接去改變值

```
protected:
    int lives;
    int money;
    int SpeedMult;
```

```
74      int GetMoney() const;
75      void EarnMoney(int money);
```

A-2 Inheritance 繼承->

```
15
16  class Enemy : public Engine::Sprite {
17  protected:
18      std::vector<Engine::Point> path;
19      float speed;
20      float hp;
21      int money;
22      PlayScene* getPlayScene();
23      virtual void OnExplode();
24  public:
25      float reachEndTime;
26      std::list<Turret*> lockedTurrets;
27      std::list<Bullet*> lockedBullets;
28      Enemy(std::string img, float x, float y, float radius, float speed, float hp, int money);
29      void Hit(float damage);
30      void UpdatePath(const std::vector<std::vector<int>>& mapDistance);
31      void Update(float deltaTime) override;
32      void Draw() const override;
33  };
34  #endif // ENEMY_HPP
35
```

```
17  NewEnemy::NewEnemy(int x, int y) : Enemy("play/enemy-7.png", x, y, 15, 70, 30, 20) {
18      if(hp < 10){
19          speed *= 2;
20      }
21      // TODO 2 (6/8): You can imitate the 2 files: 'SoldierEnemy.hpp', 'SoldierEnemy.c
22  }
23
```

A-3 Polymorphism 多型 virtual override

```
void Sprite::Update(float deltaTime) {
    Position.x += Velocity.x * deltaTime;
    Position.y += Velocity.y * deltaTime;
}
```

A-4 OOP 的好處：1.建新東西比較快，不用一直複製相同的 code 2.易維護，比較不會少改 code 3. 資料和行為分開

A-6 oop 只需要用一個指令 但如果要對同類型不同物件做特殊處理較困難

A-7 BFS 從終點做只需要一個點開始 好處理

A-8 用 image 處理 bar end 的東西 然後改變 position.x 代表 slider 改變

A-9 換幕會 delete 不需要再多寫一次 2.win-scene 名字錯誤

A-10 把 keycode pushback 進 codestroke

A-11 filename: astronomia.ogg

A-12 按 x

B-1 Icontrol->mousein->clickcallback/valuechangecallback

```
void Slider::SetOnValueChangedCallback(std::function<void(float value)> onValueChangedCallback)
    OnValueChangedCallback = onValueChangedCallback;
    // TODO 4 (2/6): Set the function pointer. Can imitate ImageButton's 'SetOnClickCallback'.
}
void Slider::SetValue(float value) {
    if(Down == true){
        this->value = value;
        OnValueChangedCallback(value);
    }
    // TODO 4 (3/6): Call 'OnValueChangedCallback' when value changed. Can imitate ImageButton'
    //               Also move the slider along the bar, to the corresponding position.
}
void Slider::OnMouseDown(int button, int mx, int my) {
    if(button & 1){
        if(mx >= End1.Position.x && mx <= End2.Position.x && mouseIn){
            Down = !Down;
        }else{
            Down = false;
        }
    }
    // TODO 4 (4/6): Set 'Down' to true if mouse is in the slider.
}
void Slider::OnMouseUp(int button, int mx, int my) {
    if(!(button & 1)) Down = false;
    // TODO 4 (5/6): Set 'Down' to false.
}
void Slider::OnMouseMove(int mx, int my) {
    if(mx >= End1.Position.x && mx <= End2.Position.x){
        Engine::ImageButton::OnMouseMove(mx, my);
        if(Down){
            float val = (mx-Bar.Position.x)/Bar.GetBitmapWidth();
            Position.x = mx;
            SetValue(val);
        }
    }
```

B-2 不能把路封死

B-3

C-1 Memory Leak 造成的原因是某個被配置的記憶體無法在被參照也無法被釋放

C-2 一塊記憶體空間只會被一個 unique_ptr 物件擁有，而不能有多個 unique_ptr 物件共用一塊記憶體空間；而當 unique_ptr 物件消失時，他所擁有的記憶體空間也就會自動被釋放掉

讓多個 shared_ptr 可以共用一份記憶體空間，並且在沒有要繼續使用的時候，可以自動把所用的資源釋放掉。而由於它的資源是可以共用的

和 shared_ptr 不同的地方在於，除了他不會增加內部的 reference counter 的計數外，它基本上也不能用來做資料的存取，主要只能用來監控 shared_ptr 目前的狀況

C-5 1. Separating Error-Handling Code from "Regular" Code 2. Grouping and Differentiating Error Types 3. don't have to check the error code after each potentially failing call 4.

C-6 std::function ->指向任一 function std::bind(….) 可以抓不同 function 的值給其他 function